After Mathematics Is Born: Algebraic Abiogenesis II, Exact Signature Refactoring, and the Optimization of Finite Algebraic Theories

Douglas C. Youvan
doug@youvan.com
www.youvan.ai
July 12, 2026

Algebraic Abiogenesis begins before a mathematical object has been named: anonymous finite operations generate term functions, recurrent equalities become candidate laws, and surviving identities induce quotient objects and algebraic theories. The present work studies the next problem. Once an algebraic theory has formed, how should its primitive language be chosen? For a finite binary term-function component C_2, a distinguished unary involution u, and a finite set $S$of binary primitives, the generated component $\langle \pi_1, \pi_2, u, S \rangle_2$ may remain unchanged while the minimum size and minimum height of representations vary substantially. We therefore define exact signature-refactoring landscapes whose vertices are content-preserving primitive signatures and whose weights measure total minimum term size, saturation size, minimum tree height, primitive width, and promotion cost. Exhaustive calculations are reported for Boolean meet-negation, Weak Kleene, Strong Kleene, and related finite components. Among 3,717 unordered binary-primitive pairs examined in the three principal systems, 868 preserve the original binary component. The optimal pair totals are 22, 36, and 190, respectively. No preserving pair is irreducibly cooperative: every successful pair contains a singleton generator. The observed gains are therefore certified macro enrichment rather than emergent joint generation. A primitive-width penalty yields exact break-even thresholds between compact vocabularies and inexpensive derivations. These results extend Algebraic Abiogenesis from the formation of theories to the exact optimization of their post-birth languages.

Keywords: Algebraic Abiogenesis, Mathematical It from Bit, abstract algebra, universal algebra, finite algebra, term functions, signature refactoring, generator spectra, term complexity, tree height, primitive promotion, clone generation, automated mathematical discovery, algebraic optimization.

A collaboration with OpenAI GPT-5.5 Thinking. CC4.0. 239 pages and 40 references.

https://orcid.org/0009-0004-4206-4253

Code for this work: https://github.com/DougYouvan/algebraic-abiogenesis-stage2

Outline

1 From Theory Birth to Theory Refactoring
1.1 Operation-First Theory Formation
1.2 The Post-Birth Problem
1.3 Content and Accessibility
1.4 Signature Refactoring
1.5 Promotion as Conservative Enrichment
1.6 Cooperative and Noncooperative Refactoring
1.7 The Refactoring Landscape

2 Blind Recovery and Structural Certification
2.1 Purpose of the Calibration Stage
2.2 Raw-Collision Selection and False Birth
2.3 Blind Cross-Order Recurrence
2.4 Frozen-Seed Reproducibility
2.5 Exhaustive Order-Three Census
2.6 Observational-Bound Stability
2.7 Structural Certification Beyond Equations
2.8 The Calibration Theorem

3 Finite Pointed Term-Function Components
3.1 Carriers, Operations, and Extensional Coding
3.2 Binary Closure
3.3 Pointed Components
3.4 Finite Saturation
3.5 Internal Generation
3.6 The Subcomponent Poset
3.7 Closure Rank
3.8 Generation and Refactoring
3.9 Relabeling Invariance
3.10 Arity-Boundedness
3.11 Finite Presentation Space
3.12 Component-Preserving Signature Space
3.13 Structural Separation

4 Exact Minimum Size and Minimum Height
4.1 Two Independent Complexity Measures
4.2 Global Profile Statistics
4.3 Size and Height Are Not Interchangeable

4.4 Extensional Dynamic Programming
4.5 Weighted Derivation Hypergraph
4.6 Monotone Finalization
4.7 Correctness of Minimum-Size Computation
4.8 Correctness of Minimum-Height Computation
4.9 Witness Terms
4.10 Independent Oracles
4.11 Saturation
4.12 Diameter Interpretation
4.13 Dominance Between Signatures
4.14 Primitive Width and Trivial Compression
4.15 Exact Promotion Gain
4.16 Exactness Under Ties
4.17 Complexity Bounds
4.18 Verification Conditions
4.19 Corrected Size-Height Semantics
4.20 Formation Profiles

5 Presentation-Dependent Formation Profiles
5.1 Fixed Content, Variable Access
5.2 Involutive Polarity Transformations
5.3 Cost of Polarity Translation
5.4 Polarity-Orbit Formation Spectrum
5.5 Cycle 7 Exact Profile Calculation
5.6 Boolean Formation Profiles
5.7 Weak Kleene Formation Profiles
5.8 Strong Kleene Formation Profiles
5.9 The BCK-Related Union Component
5.10 Profile Stratification
5.11 Relative Accessibility
5.12 Primitive Centrality
5.13 Presentation Entropy
5.14 Profile Equivalence
5.15 Conjugacy and Profile Invariance
5.16 Formation Profiles as Algebraic Data
5.17 Cycle 7 Result

6 Genuine and False-Birth Complexity Spectra
6.1 Complexity After Structural Certification

6.2 Equational Agreement and Structural Failure
6.3 Quotient Repair
6.4 Complexity of a False Birth
6.5 Genuine and False-Birth Spectra
6.6 The BCK-Related Union Spectrum
6.7 Certificate Layers
6.8 Negative Certificates as Mathematical Output
6.9 Explanation-Subtracted Complexity
6.10 Local and Global Macros
6.11 Spectrum Distance
6.12 False-Birth Neighborhoods
6.13 Birth Validity and Presentation Quality
6.14 No Complexity Threshold for Validity
6.15 Genuine Formation Spectra
6.16 False-Birth Spectra as Countertheory Data
6.17 Cycle 8 Result

7 Recursive Primitive Promotion
7.1 Conservative Enrichment
7.2 Pointwise Monotonicity
7.3 Direct and Propagated Gain
7.4 Promotion Ranking
7.5 Recursive Promotion Sequence
7.6 Termination
7.7 Greedy Optimality and Its Limits
7.8 Diminishing Returns
7.9 Promotion Graph
7.10 Promotion Under Primitive Cost
7.11 Break-Even Price of a Macro
7.12 Height-Aware Promotion
7.13 Promotion Witnesses
7.14 Recursive Promotion as Language Formation
7.15 Promotion and Abbreviation
7.16 Reversibility
7.17 Promotion Depth
7.18 Path Dependence
7.19 Cycle 9 Computation

7.20 Promotion Theorem
7.21 Interpretation of Cycle 9

8 Full Single-Generator Birth Spectra
8.1 From Selected Orbits to Exhaustive Generator Sets
8.2 Exhaustive Generator Test
8.3 Generator Density
8.4 Exact Generator Counts
8.5 Equality with the Polarity Orbit
8.6 Orbit Sufficiency
8.7 Profile Equivalence Classes
8.8 Boolean Full-Generator Spectrum
8.9 Weak Kleene Full-Generator Spectrum
8.10 Strong Kleene Full-Generator Spectrum
8.11 Profile Multiplicity and Symmetry
8.12 Complete Singleton Pareto Sets
8.13 Generator Spectrum as an Isomorphism Invariant
8.14 Spectrum Hashes and Reproducibility
8.15 Completeness Relative to Arity Two
8.16 Scope of the Larger Components
8.17 Discovery Significance
8.18 Cycle 10 Theorem

9 Exact Signature-Refactoring Landscapes
9.1 Width-Two Presentation Space
9.2 Signature States
9.3 Promotion and Demotion Edges
9.4 Edge Weights
9.5 Exact Landscape Census
9.6 Graph Sizes
9.7 Original, Singleton, Anchored-Pair, and Global-Pair Optima
9.8 Boolean Landscape
9.9 Weak Kleene Landscape
9.10 Strong Kleene Landscape
9.11 Aggregate Reduction Theorem
9.12 Pareto Structure
9.13 Shortest Preserving Refactoring Paths
9.14 Stepwise Access Changes
9.15 Direct Anchored Promotion Versus Global Refactoring

9.16 Refactoring Distance
9.17 Landscape Connectedness
9.18 Exactness and Reproducibility
9.19 Exact Signature-Landscape Theorem

10 Cooperative Generation and Its Absence
10.1 Two Mechanisms of Pair Completeness
10.2 Minimal Generating Signatures
10.3 Lattice Formulation
10.4 Closure-Rank Formulation
10.5 Macro Enrichment
10.6 Exhaustive Null Result
10.7 Consequence for Minimal Generators
10.8 Pair Count Predicted by Singleton Incidence
10.9 Preserving-Pair Density Formula
10.10 Structure of the Width-Two Graph
10.11 Connectedness Revisited
10.12 Negative Results as Discovery Results
10.13 Why Cooperation Would Have Been Stronger
10.14 Cooperation Defect
10.15 Cooperative Depth
10.16 Cooperative Search Beyond Existing Components
10.17 Search Policy for Pair-Born Components
10.18 Cooperation Versus Complementarity
10.19 Cooperation and Primitive Cost
10.20 Scientific Interpretation
10.21 Cooperative Null Corollary for the Best Pairs
10.22 Theorem of Bounded Generative Nonemergence
10.23 Research Consequence

11 Penalized Vocabulary and Optimal Presentation
11.1 The Free-Primitive Problem
11.2 Interpretation of the Primitive Price
11.3 Width-Constrained Optima
11.4 Break-Even Threshold
11.5 Piecewise-Linear Lower Envelope
11.6 Multiplicity of Optimal Signatures
11.7 Primitive Price and Macro Utility
11.8 Original-Anchored Thresholds

11.9 Refactoring Cost Versus Steady-State Cost
11.10 Amortization of the Global Pair Refactorings
11.11 Height-Penalized Optimization
11.12 Strong Kleene Total-Size Versus Radius Tradeoff
11.13 Boolean and Weak Kleene Height Tradeoffs
11.14 Pair Versus Low-Height Singleton
11.15 Primitive Description Length
11.16 Minimal Description Presentation
11.17 Robust Optimization
11.18 Sensitivity to the Width Bound
11.19 Concavity of the Optimal Value
11.20 Optimality Certificates
11.21 The Exact Optimization Theorem
11.22 Interpretation
11.23 From Theory Formation to Variational Language

12 Reproducibility, Scope, and Open Problems
12.1 Exact Finite Evidence
12.2 Repository Structure
12.3 Manifest Verification
12.4 Reproduction and Self-Certification
12.5 Independent Recalculation
12.6 Test Coverage
12.7 Sharded Exhaustive Computation
12.8 Computational Scope
12.9 Binary-Component Limitation
12.10 Finite Model Limitation
12.11 Historical Novelty Limitation
12.12 Interpretation Limitation
12.13 Policy Dependence
12.14 Metric Dependence
12.15 Tree Terms Versus Shared Circuits
12.16 Target-Free Circuit Discovery
12.17 Search for Cooperative Components
12.18 Higher-Width Minimal Generators
12.19 Higher-Arity Refactoring
12.20 Sparse Closure Methods
12.21 Symmetry Reduction

12.22 Proof Extraction
12.23 Formal Verification
12.24 Automated Literature Equivalence
12.25 Preregistered Benchmarks
12.26 Novelty Standard
12.27 Toward a General Refactoring Functor
12.28 Toward Variational Objecthood
12.29 What Has Been Demonstrated
12.30 What Has Not Been Demonstrated
12.31 The Next Decisive Result
12.32 Conclusion

References

1. From Theory Birth to Theory Refactoring

1.1 Operation-First Theory Formation

Algebraic Abiogenesis begins with a finite carrier $A$and one or more anonymous operations on A. No interpretation is assigned to the operations, no target variety is supplied, and no preferred identities are presupposed. Terms are formed syntactically and evaluated extensionally on all variable assignments. Distinct terms $s$and $t$collide when they induce the same term function:

$$s^A = t^A.$$

Each collision is exact. It is an equality of finite functions, not a statistical similarity between sampled outputs.

A single collision need not reveal transferable structure. It may arise from a constant operation, a projection, a small image, a low-order accident, or a consequence of a simpler identity. The Stage I discovery process therefore separates raw collision generation from theory formation. Candidate identities are compared across independently produced models, filtered for degeneracy, tested against countermodels, and reduced by explanation subtraction. The resulting progression is

$$\text{anonymous operation} \longrightarrow \text{term-function collisions} \longrightarrow \text{recurrent identities} \longrightarrow \text{axiomatic compression} \longrightarrow \text{quotient theory}.$$

When a set $E$of equations is retained, it induces a congruence on the term algebra $T_\Sigma(X)$:

$$s \equiv_E t \Leftrightarrow E \vdash s \approx t.$$

The quotient

$$F_E(X) = T_\Sigma(X)/\equiv_E$$

is the free algebra determined by the discovered equational theory. Formal expressions that were initially distinct become representatives of the same algebraic object.

Theory birth is therefore a passage from syntactic multiplicity to certified equivalence:

$$\text{syntax} \longrightarrow \text{equivalence} \longrightarrow \text{quotient} \longrightarrow \text{object}.$$

Stage I established that this passage can be implemented reproducibly in finite experimental settings. It did not require that the resulting theories be historically new. Recovery of a known theory remains evidence that an operation-first procedure can reconstruct coherent mathematics without being given the name, axioms, or intended interpretation of the structure in advance.

1.2 The Post-Birth Problem

The formation of a theory does not determine a unique primitive vocabulary.

Suppose a finite algebraic component has been generated from a unary operation $u$and a binary operation b. A derived binary function $c$may itself generate the same component when used in place of b. Alternatively, $c$may be added to the original signature as a new primitive. In both cases, the extensionally available binary functions may remain unchanged while their minimum representations change.

Thus theory formation leaves a second problem unresolved:

Which primitive signature should present the born theory?

Classical universal algebra commonly identifies signatures that generate the same term operations as equivalent for a specified purpose. Such equivalence preserves operational content but suppresses differences in accessibility. A function represented by a term of size ten in one signature may be primitive in another. A deeply nested construction in one presentation may have height two in another. Two term-equivalent signatures may therefore differ substantially as generative architectures.

The present stage treats primitive choice as an exact optimization problem.

Let Abe finite. Let

$$\pi_1(x, y) = x, \pi_2(x, y) = y$$

be the binary projections, and let $u: A \to A$be a fixed unary operation. For a finite set $S$of binary operations on A, define

$$C_2(S; u) = \langle \pi_1, \pi_2, u, S \rangle_2,$$

the least set of binary functions containing the projections and closed under the action of $u$and every operation in S.

The subscript 2is essential. Only binary term functions are considered. No claim of unrestricted clone equality follows from equality of binary components alone.

Definition 1.1. A pointed binary term-function component is a pair

$$(C_2, u),$$

where $C_2 \subseteq A^{A^2}$is finite, contains π_1, π_2, and is closed under the fixed unary operation $u$and the relevant binary compositions.

Definition 1.2. A finite binary signature $S \subseteq C_2$is preserving when

$$C_2(S; u) = C_2.$$

The set of preserving signatures is

$$\mathcal{R}(C_2, u) = \{S \subseteq C_2 : C_2(S; u) = C_2\}.$$

Every $S \in \mathcal{R}(C_2, u)$presents the same binary operational content. The purpose of the refactoring analysis is to determine how those presentations differ.

1.3 Content and Accessibility

A term-function component records which binary functions are generable. It does not, by itself, record the cost of generating them from a chosen primitive vocabulary.

For a signature S, let $s_S(f)$be the minimum number of operation nodes in a term representing f. Variables have size zero:

$$s(x) = s(y) = 0.$$

Unary application contributes one node:

$$s(u(t)) = s(t) + 1.$$

Binary application contributes one node in addition to the sizes of both arguments:

$$s(b(t_1, t_2)) = s(t_1) + s(t_2) + 1.$$

Define the total size cost

$$T(S) = \sum_{f \in C_2} s_S(f)$$

and the saturation size

$$S_{\max}(S) = \max_{f \in C_2} s_S(f).$$

Tree height is defined independently:

$$h(x) = h(y) = 0,$$
$$h(u(t)) = h(t) + 1,$$
$$h(b(t_1, t_2)) = \max\{h(t_1), h(t_2)\} + 1.$$

For $f \in C_2$, let $h_S(f)$be the minimum height of a term representing f, and define

$$H_{\max}(S) = \max_{f \in C_2} h_S(f).$$

These quantities assign an access geometry to the presentation S.

Proposition 1.1. Let $S, S' \in \mathcal{R}(C_2, u)$. Then

$$C_2(S; u) = C_2(S'; u) = C_2$$

does not imply

$$s_S(f) = s_{S'}(f)$$

or

$$h_S(f) = h_{S'}(f)$$

for every $f \in C_2$.

Proof. Let $m \in C_2$be nonprimitive relative to S, with

$$s_S(m) > 1.$$

Set

$$S' = S \cup \{m\}.$$

Since $m \in C_2(S; u)$,

$$C_2(S'; u) = C_2(S; u).$$

However, $m$is primitive in S', so

$$s_{S'}(m) = 1 < s_S(m).$$

An analogous argument applies to height whenever

$$h_S(m) > 1.$$

Therefore preservation of generated content does not preserve minimum representation profiles. □

The proposition separates two layers of algebraic description:

extensional content

and

presentation-relative accessibility.

The first is invariant under preserving signature change. The second is not.

1.4 Signature Refactoring

The term refactoring is used for a change of primitive signature that preserves the designated binary component.

Definition 1.3. A signature refactoring of $(C_2, u)$is a transition

$$S \dashrightarrow S'$$

with

$$S, S' \in \mathcal{R}(C_2, u).$$

A refactoring may add, remove, or replace binary primitives, provided that the generated binary component remains equal to C_2.

The preservation condition supplies the semantic invariant:

$$C_2(S;u) = C_2(S';u).$$

The complexity profiles supply the changing quantities:

$$T(S), S_{\max}(S), H_{\max}(S).$$

A refactoring is size-improving when

$$T(S') < T(S),$$

radius-improving when

$$S_{\max}(S') < S_{\max}(S),$$

and height-improving when

$$H_{\max}(S') < H_{\max}(S).$$

The criteria need not agree. One signature may minimize total size while another minimizes maximum size or tree height.

Consequently, there need not be a unique best presentation independent of objective. Primitive selection is multiobjective.

A natural signature descriptor is

$$\Phi(S) = (\mid S \mid, T(S), S_{\max}(S), H_{\max}(S)).$$

The first coordinate measures vocabulary width. The remaining coordinates measure derivational cost. Refactoring therefore exposes a tradeoff between primitive economy and access economy.

1.5 Promotion as Conservative Enrichment

Let $S \in \mathcal{R}(C_2, u)$, and let $m \in C_2$. Promoting $m$forms the enlarged signature

$$S^+ = S \cup \{m\}.$$

Because $m$was already derivable,

$$C_2(S^+; u) = C_2(S; u).$$

Promotion is therefore conservative with respect to binary extensional content.

Proposition 1.2. For every $f \in C_2$,

$$s_{S^+}(f) \leq s_S(f)$$

and

$$h_{S^+}(f) \leq h_S(f).$$

Proof. Every term available in Sremains available in S^+. The enlarged signature may introduce additional representations but removes none. Therefore the minima cannot increase. □

Define the total promotion gain

$$\Delta_T(S, m) = T(S) - T(S \cup \{m\}),$$

the saturation-size gain

$$\Delta_{\max}(S, m) = S_{\max}(S) - S_{\max}(S \cup \{m\}),$$

and the saturation-height gain

$$\Delta_H(S, m) = H_{\max}(S) - H_{\max}(S \cup \{m\}).$$

Promotion converts a derived function into a primitive macro. The resulting cost reduction may extend far beyond the promoted function itself because other terms can reuse the new primitive.

The operation is not theory birth in the Stage I sense. No new binary function is created. The change occurs in the organization of access to already existing functions.

1.6 Cooperative and Noncooperative Refactoring

A preserving pair $\{g, h\}$may arise in two structurally different ways.

First, one member may already generate C_2:

$$C_2(\{g\}; u) = C_2.$$

The second member then acts as a macro. The pair is preserving because it contains a complete singleton generator.

Second, neither member may be complete alone:

$$C_2(\{g\}; u) \neq C_2,$$
$$C_2(\{h\}; u) \neq C_2,$$

while jointly

$$C_2(\{g, h\}; u) = C_2.$$

The second case is irreducible cooperation.

Definition 1.4. A preserving pair $\{g, h\}$is cooperative when

$$C_2(\{g\}; u) \neq C_2, C_2(\{h\}; u) \neq C_2,$$

but

$$C_2(\{g, h\}; u) = C_2.$$

Cooperative generation would represent a stronger post-birth phenomenon than primitive promotion. Completeness would arise from interaction between two individually incomplete primitives.

The exhaustive Boolean, Weak Kleene, and Strong Kleene width-two landscapes reported below contain no cooperative pair. Every preserving pair contains at least one singleton generator.

The observed improvements are therefore classified as

$$\text{complete basis} + \text{promoted macro},$$

not as

$$\text{incomplete fragment} + \text{incomplete fragment} \longrightarrow \text{complete component.}$$

This negative result defines the boundary of the current achievement. Signature refactoring has been demonstrated exactly. Cooperative signature formation has not.

1.7 The Refactoring Landscape

The preserving-signature family may be treated as a finite weighted state space.

For a fixed width bound k, define

$$\mathcal{R}_{\leq k}(C_2, u) = \{S \in \mathcal{R}(C_2, u) : 1 \leq |\, S \,| \leq k\}.$$

Vertices are preserving signatures. An edge may connect $S$and $S'$when they differ by one primitive addition, removal, or replacement. Each vertex is weighted by $\Phi(S)$.

This yields the refactoring graph

$$\mathbf{R}_k(C_2, u).$$

Paths in $\mathbf{R}_k(C_2, u)$encode sequences of content-preserving signature changes. A path may be optimized for the number of primitive edits, cumulative representation cost, maximum intermediate cost, or vocabulary width.

The born theory is therefore associated not with one canonical language but with a finite ecology of equivalent primitive presentations.

The governing optimization problem is

$$S_\lambda^* \in \underset{S \in \mathcal{R}_{\leq k}(C_2, u)}{\arg\min} \, [T(S) + \lambda(|\, S \,| - 1)].$$

The parameter $\lambda \geq 0$assigns a price to each primitive beyond the first. For small λ, a wider vocabulary may be preferred because it reduces derivational cost. For large λ, a compact singleton basis may be optimal.

The equation formalizes the transition from theory birth to theory design.

Stage I asks:

$$\text{Which equivalences make an algebraic object visible?}$$

Stage II asks:

Which preserving signature makes that object most accessible?

The complete methodological progression is

finite information ⟶ identity ⟶ quotient ⟶ theory ⟶ refactoring landscape ⟶ optimized presentation.

2. Blind Recovery and Structural Certification

2.1 Purpose of the Calibration Stage

Before optimizing the primitive language of a finite algebraic theory, the discovery process must establish that it can recover coherent structure without being supplied with the target theory. Otherwise, signature optimization risks becoming only a retrospective analysis of familiar algebras selected by hand.

The initial Stage II cycles therefore imposed a blind calibration problem. Anonymous finite operations were evaluated under fixed search policies. The system was required to distinguish among:

degenerate collision abundance,
locally fertile but nontransferable structure,
recurrent known algebraic families,

and

apparently coherent but structurally inadmissible theories.

The resulting workflow was not designed to maximize the number of equations reported. It was designed to maximize the probability that a reported birth survived successive forms of rejection.

Let $\mathcal{A}_n$ denote a finite population of labeled algebras on an n-element carrier in a fixed signature. For $A \in \mathcal{A}_n$, let

$$\mathrm{Eq}_d(A)$$

be the set of term identities detected within a finite observational universe of terms of depth or size bounded by d.

A raw collision score has the general form

$$R_d(A) = | \operatorname{Eq}_d(A) |.$$

Such a score is insufficient. Constant operations, projections, low-image operations, and other degeneracies can produce very large identity sets for mathematically shallow reasons.

The calibration stage therefore replaces raw fertility with filtered and explanation-subtracted fertility.

If E is a set of already recognized identities, define the residual identity set

$$\operatorname{Res}_d(A \mid E) = \operatorname{Eq}_d(A) \setminus \operatorname{Cn}_d(E),$$

where $\operatorname{Cn}_d(E)$ is the set of identities within the observational universe that are consequences of E.

The residual score is

$$F_d(A \mid E) = | \operatorname{Res}_d(A \mid E) |.$$

The operative question is not

How many equalities occur?

but

How much unexplained regularity remains?

This distinction governs the blind-recovery experiments.

2.2 Raw-Collision Selection and False Birth

The first calibration exposed a basic failure mode. If candidate systems are ranked only by the number of exact term-function collisions, the highest-scoring systems may be structurally ineligible.

Let $\mathcal{D} \subseteq \mathcal{A}_n$ denote the degenerate region determined by declared filters. Typical exclusions include operations that are constant, projections, essentially unary, excessively low-image, or otherwise unable to support the intended level of algebraic interaction.

A raw winner is

$$A^*_{\mathrm{raw}} \in \arg\max_{A \in \mathcal{A}_n} R_d(A).$$

In the exhaustive order-two experiment, the raw maximum was attained only by systems rejected by the admissibility conditions:

$$A^*_{\mathrm{raw}} \in \mathcal{D}.$$

Thus collision abundance alone selected false births.

Proposition 2.1. Raw term-function collision count is not a reliable proxy for algebraic fertility.

Proof. Consider a constant binary operation $c(x, y) = a$. Every nonvariable term containing $c$evaluates to the same constant function. Hence a large syntactic term population collapses into a very small number of term functions, producing many exact collisions. The resulting collision count may exceed that of a richer operation while encoding only constant collapse. Similar arguments apply to projections and essentially unary operations. Therefore collision abundance does not imply nondegenerate algebraic structure. □

The false-birth result was retained as evidence rather than removed as an inconvenience. It demonstrated that the discovery system required explicit anti-degeneracy conditions before any claim of theory formation could be made.

The corrected candidate region is

$$\mathcal{A}_n^{\mathrm{adm}} = \mathcal{A}_n \setminus \mathcal{D}.$$

All subsequent rankings were restricted to admissible systems or assigned penalties sufficient to prevent degenerate domination.

2.3 Blind Cross-Order Recurrence

An identity holding in one finite algebra may be accidental. Transferability is strengthened when the same identity family occurs across independently generated systems and across different carrier orders.

Let $A \in \mathcal{A}_m$and $B \in \mathcal{A}_n$, with $m \neq n$. Define the common observational theory

$$\mathrm{Com}_d(A, B) = \mathrm{Eq}_d(A) \cap \mathrm{Eq}_d(B).$$

For a family $\mathcal{F}$ of finite algebras, define

$$\mathrm{Com}_d(\mathcal{F}) = \bigcap_{A \in \mathcal{F}} \mathrm{Eq}_d\,(A).$$

A candidate law is strengthened when it survives enlargement of $\mathcal{F}$, particularly when the family contains multiple carrier orders and independently generated tables.

Blind cross-order recurrence recovered systems equivalent to familiar Boolean, Weak Kleene, Strong Kleene, and BCK-related structures. The target names were not supplied to the search. They were assigned only after the finite operational behavior had been recovered and structurally interpreted.

This distinction is essential:

$$\text{blind recovery} \neq \text{historical novelty.}$$

The recovered systems were new to the search process but not new to mathematics.

Definition 2.1. A finite theory birth is blind relative to a target class $\mathcal{V}$ when the search policy does not include the axioms, names, defining operations, or canonical finite models of $\mathcal{V}$ as target information.

Blindness does not require the absence of all human design. Carrier sizes, operation arities, term bounds, degeneracy filters, and ranking policies remain externally specified. It requires only that the target algebraic family not be encoded as the desired answer.

The recovery of named structures therefore served as calibration of target-free recognition rather than as a novelty claim.

2.4 Frozen-Seed Reproducibility

Evolutionary or stochastic search can produce attractive isolated examples that disappear under changes of random seed. To test whether the recovery process was reproducible, twenty seed-controlled experiments were frozen before interpretation.

Let

$$\sigma_1, \ldots, \sigma_{20}$$

be the declared random seeds. For each seed σ_i, the complete computational path produced a finite record

$$\mathcal{E}_i = (\mathcal{P}_i, \mathcal{T}_i, \mathcal{C}_i, \mathcal{M}_i),$$

where $\mathcal{P}_i$is the generated population, $\mathcal{T}_i$the observational term universe, $\mathcal{C}_i$the candidate ranking, and $\mathcal{M}_i$the final manifests and checksums.

The principal reproducibility criterion was not that every seed return the same named algebra. The criterion was that, whenever the relevant family occurred in the generated population and survived the declared filters, the fixed analysis pipeline classified it consistently.

Proposition 2.2. Under frozen term bounds, filters, scoring rules, and tie-breaking policies, the finite analysis of a generated population is deterministic.

Proof. Every term is evaluated on every assignment in a fixed finite carrier. Equality of term functions is exact. Candidate scores, residual sets, isomorphism reductions, and certificate tests are finite deterministic computations. Once the generated operation tables and policy parameters are fixed, the resulting classifications are fixed. □

Randomness therefore enters population generation, not finite verification.

This separation permits the evidentiary record to distinguish:

stochastic proposal

from

deterministic certification.

The frozen-seed runs showed that the method did not depend on one unrecorded favorable trajectory. They did not establish asymptotic convergence or universal recovery probabilities. They established reproducible behavior on a declared finite benchmark.

2.5 Exhaustive Order-Three Census

The strongest calibration removed stochastic population generation entirely.

Consider a carrier

$$A = \{0,1,2\}$$

with one fixed unary operation $u$and one variable binary operation b. A binary table has nine entries, each taking one of three values. Hence the complete labeled binary-operation population has size

$$3^9 = 19{,}683.$$

The full labeled unary-binary population depends on the allowed unary tables. In the exhaustive census implemented in the release, the declared parameterization produced

$$531{,}441$$

labeled order-three systems.

Every one of these systems was evaluated under the same finite search and admissibility policy.

After the declared filters, the surviving population contained

$$83{,}638$$

labeled systems.

Carrier relabeling was then used to quotient labeled duplicates. If $\rho \in S_3$is a permutation of the carrier, the relabeled operations are

$$u^{\rho} = \rho \circ u \circ \rho^{-1}$$

and

$$b^{\rho}(x, y) = \rho\,(b(\rho^{-1}(x), \rho^{-1}(y))).$$

Two systems are isomorphic when they lie in the same S_3-orbit.

The eligible labeled systems collapsed to

$$13{,}965$$

isomorphism classes.

Theorem 2.1. Under the frozen Stage II census policy, the complete order-three search space comprised $531{,}441$labeled systems, of which $83{,}638$survived the admissibility filters and represented $13{,}965$eligible isomorphism classes.

The theorem is finite and policy-relative. It does not claim that $13{,}965$is an invariant of all possible admissibility definitions. It certifies the complete census under the declared one.

The exhaustive calculation served three purposes.

First, it tested whether evolutionary search had overlooked globally superior systems.

Second, it established that candidate ranking could be compared against a complete finite population.

Third, it enabled exact statements about the stability of the selected winner as the observational term universe expanded.

2.6 Observational-Bound Stability

A finite term universe gives only a bounded view of an algebra's equational behavior. A candidate that ranks highly under a small term set may fall when larger or deeper terms are included.

Let

$$\mathcal{T}_1 \subset \mathcal{T}_2$$

be two observational term universes. In the release, the relevant bounds contained

$$|\ \mathcal{T}_1\ | = 224$$

and

$$|\ \mathcal{T}_2\ | = 1{,}514$$

terms.

For an eligible isomorphism class $[A]$, let

$$Q_{\mathcal{T}}([A])$$

be the frozen discovery score computed from identities observed in $\mathcal{T}$, after degeneracy filtering and explanation subtraction.

Weak Kleene was the global winner at both observational bounds:

$$[A_{\mathrm{WK}}] \in \arg\max_{[A]} Q_{\mathcal{T}_1}([A])$$

and

$$[A_{\mathrm{WK}}] \in \arg\max_{[A]} Q_{\mathcal{T}_2}([A]).$$

Proposition 2.3. The Weak Kleene winner was stable under expansion from the 224-term observational universe to the 1,514-term universe.

This is a finite robustness statement. It does not prove that Weak Kleene remains optimal under all term bounds or all scoring policies. It establishes that the selected result was not an artifact of only the smaller frozen observational window.

The persistence of the winner also demonstrates that blind recovery can favor a coherent known theory over large populations of accidental finite regularities.

2.7 Structural Certification Beyond Equations

Equational recurrence alone can identify an apparent theory whose intended non-equational structure fails.

The BCK-related calibration exposed this distinction. A finite system reproduced the bounded equational behavior associated with a target family but failed additional structural requirements.

Let a binary operation $*$and distinguished element 0induce a relation

$$x \preceq y \Leftrightarrow x * y = 0.$$

For the intended ordered interpretation, antisymmetry requires

$$x \preceq y \ \wedge \ y \preceq x \Longrightarrow x = y.$$

Equational agreement within a finite observational universe does not guarantee this property.

The candidate duplicate failed the antisymmetry test. There existed distinct $x, y \in A$such that

$$x * y = 0$$

and

$$y * x = 0.$$

Hence the induced relation was not a partial order.

A natural repair would identify mutually related elements:

$$x \sim y \Leftrightarrow x \preceq y \ \wedge \ y \preceq x.$$

For the quotient $A/\sim$to inherit the operation, $\sim$must be compatible with $*$:

$$x \sim x', y \sim y' \Longrightarrow x * y \sim x' * y'.$$

The candidate failed this quotient-compatibility condition as well.

Therefore neither the original system nor the proposed quotient supplied the intended certified object.

Definition 2.2. A structural birth certificate is a finite record containing:

the candidate identities,
their exact finite witnesses,
required non-equational properties,
countermodel tests,
quotient-compatibility tests,

and

the final admissibility decision.

The BCK-related candidate received a negative certificate.

Proposition 2.4. Bounded equational agreement does not entail validity of an intended ordered or quotient structure.

Proof. The finite candidate satisfied the tested equational conditions but failed antisymmetry. Its natural identification relation also failed compatibility with the operation. Thus the equations did not suffice to certify either the intended ordered algebra or a well-defined quotient repair. □

This result prevents an equation-rich duplicate from being reported as a successful birth merely because its bounded identity profile resembles that of a known theory.

2.8 The Calibration Theorem

The first six cycles collectively establish the following finite methodological result.

Theorem 2.2. Under the frozen Stage II policies, the Algebraic Abiogenesis pipeline:

rejects raw-collision false births,
recovers named algebraic families without target identities,
reproduces its finite classifications under frozen seeds,

exhaustively ranks the declared order-three population,
preserves its Weak Kleene winner under a substantial expansion of the term universe,

and

rejects an equation-matching candidate that fails structural certification.

The theorem does not assert that the pipeline discovers historically new mathematics. It establishes that the system can perform several tasks required before such a claim would become credible:

blind recovery,
finite global comparison,
false-positive rejection,
structural certification,

and

reproducible evidence preservation.

These calibrations justify the transition to exact signature analysis.

Once a candidate theory has survived the birth process, a new class of questions becomes available. One may hold the extensional binary component fixed and vary the primitive presentation. The search then moves from

Which theory has formed?

to

How is the formed theory most efficiently generated?

The remaining sections study that post-birth problem.

3. Finite Pointed Term-Function Components

3.1 Carriers, Operations, and Extensional Coding

Let Abe a finite carrier with

$$| A | = q \geq 2.$$

For each arity $n \geq 1$, let

$$\mathcal{O}_A^{(n)} = A^{A^n}$$

denote the set of all n-ary operations on A. Its cardinality is

$$| \mathcal{O}_A^{(n)} | = q^{q^n}.$$

The present work studies the binary operation space

$$\mathcal{O}_A^{(2)} = A^{A^2}.$$

Every $f \in \mathcal{O}_A^{(2)}$is represented extensionally by its complete value vector on a fixed ordering of A^2:

$$\mathrm{tab}(f) = (f(a_1, b_1), \ldots, f(a_{q^2}, b_{q^2})).$$

Because $A$is finite,

$$f = g \Leftrightarrow \mathrm{tab}(f) = \mathrm{tab}(g).$$

Thus operation equality is decidable by finite table comparison.

The binary projections are

$$\pi_1(x, y) = x, \pi_2(x, y) = y.$$

A unary operation $u: A \to A$acts on binary functions pointwise:

$$u_*(f)(x, y) = u(f(x, y)).$$

A binary operation $b: A^2 \to A$acts on pairs of binary functions by superposition:

$$b_*(f, g)(x, y) = b(f(x, y), g(x, y)).$$

The distinction between the carrier-level operation $b$and its induced operation b_*on binary term functions will usually be suppressed when no ambiguity arises.

3.2 Binary Closure

Let $S \subseteq \mathcal{O}_A^{(2)}$be a finite set of binary primitives, and let $U \subseteq \mathcal{O}_A^{(1)}$be a finite set of unary primitives.

Define

$$\mathrm{Cl}_2(U, S)$$

to be the least subset $C \subseteq \mathcal{O}_A^{(2)}$satisfying:

$$\pi_1, \pi_2 \in C,$$
$$f \in C,\ u \in U \Longrightarrow u \circ f \in C,$$

and

$$f, g \in C,\ b \in S \Longrightarrow b(f, g) \in C.$$

Equivalently,

$$\mathrm{Cl}_2(U, S) = \bigcap \left\{ C \subseteq \mathcal{O}_A^{(2)} : C \text{ satisfies the three closure conditions} \right\}.$$

When $U = \{u\}$, write

$$C_2(S; u) = \mathrm{Cl}_2(\{u\}, S).$$

When $S = \{b\}$, write

$$C_2(u, b) = C_2(\{b\}; u).$$

The resulting object is the binary term-function component generated by the signature.

Proposition 3.1. The operator

$$S \longmapsto C_2(S; u)$$

is extensive, monotone, and idempotent in the following sense:

$$S \subseteq C_2(S;u),$$
$$S \subseteq T \Longrightarrow C_2(S;u) \subseteq C_2(T;u),$$

and

$$C_2(C_2(S;u);u) = C_2(S;u).$$

Proof. Extensivity follows because every primitive is admitted at the first stage of closure. Monotonicity follows because every construction available from Sremains available from $T \supseteq S$. Idempotence follows because $C_2(S;u)$is already closed under the projections, the action of u, and binary superposition by all of its members. □

The closure operator partitions primitive sets according to the binary component they generate.

Definition 3.1. Two finite binary signatures $S, T \subseteq \mathcal{O}_A^{(2)}$are binary-term equivalent relative to $u$when

$$C_2(S;u) = C_2(T;u).$$

Write

$$S \equiv_{2,u} T.$$

This is an equivalence relation on finite binary signatures.

3.3 Pointed Components

The unary operation $u$remains fixed throughout each refactoring landscape. It therefore acts as a distinguished point in the generative environment.

Definition 3.2. A finite pointed binary term-function component is a pair

$$(C, u)$$

such that:

1. $C \subseteq \mathcal{O}_A^{(2)}$;
2. $\pi_1, \pi_2 \in C$;
3. $u \circ f \in C$for every $f \in C$;

4. $b(f,g) \in C$ for every $b, f, g \in C$.

The fourth condition makes C closed under binary superposition by its own members. In the computational instances,

$$C = C_2(S; u)$$

for some finite primitive set S.

The pointed structure separates the fixed unary background from the variable binary presentation. This is useful when one wishes to compare several binary bases under a common involution or negation.

The fixed unary operation may itself satisfy additional conditions. In the principal examples,

$$u^2 = \mathrm{id}_A.$$

Thus u is an involution:

$$u(u(x)) = x.$$

The refactoring framework does not require involutivity in general, but the Boolean and Kleene examples use it.

3.4 Finite Saturation

Because

$$C_2(S; u) \subseteq \mathcal{O}_A^{(2)}$$

and $\mathcal{O}_A^{(2)}$ is finite, iterative closure terminates.

Define

$$C^{(0)} = \{\pi_1, \pi_2\} \cup S.$$

Inductively, let

$$C^{(k+1)} = C^{(k)} \cup \{u \circ f : f \in C^{(k)}\} \cup \{b(f,g) : b \in S,\ f, g \in C^{(k)}\}.$$

Then

$$C^{(0)} \subseteq C^{(1)} \subseteq C^{(2)} \subseteq \cdots .$$

Since there are at most

$$q^{q^2}$$

binary functions, there exists Nsuch that

$$C^{(N)} = C^{(N+1)}.$$

At that point,

$$C^{(N)} = C_2(S; u).$$

Proposition 3.2. The iterative closure process stabilizes after at most

$$q^{q^2} - | C^{(0)} |$$

strict enlargement steps.

Proof. Each strict enlargement adds at least one previously absent binary function. No more than q^{q^2} distinct binary functions exist. □

This bound is crude but sufficient for decidability.

Corollary 3.1. For finite A, the following questions are decidable:

$$f \in C_2(S; u),$$
$$C_2(S; u) = C_2(T; u),$$

and

$$C_2(S; u) = C$$

for any explicitly tabulated f, S, T, C.

The exact refactoring analysis depends on this finite decidability.

3.5 Internal Generation

Once a reference component C has been computed, one may ask which subsets of C regenerate it.

For $S \subseteq C$, define the internal closure

$$\langle S \rangle_{C,u} = C_2(S; u).$$

Since every primitive in S already belongs to C, monotonicity gives

$$C_2(S; u) \subseteq C.$$

Definition 3.3. A subset $S \subseteq C$ is a generating signature for (C, u) when

$$C_2(S; u) = C.$$

The family of all generating signatures is

$$\mathrm{Gen}(C, u) = \{S \subseteq C : C_2(S; u) = C\}.$$

The width-k generating family is

$$\mathrm{Gen}_k(C, u) = \{S \in \mathrm{Gen}(C, u) : | S | = k\}.$$

The bounded family is

$$\mathrm{Gen}_{\leq k}(C, u) = \bigcup_{j=1}^{k} \mathrm{Gen}_j (C, u).$$

The Stage 2 release computes

$$\mathrm{Gen}_1(C, u)$$

and

$$\mathrm{Gen}_2(C, u)$$

for the principal Boolean and Kleene components.

Proposition 3.3. The generating family is upward closed within 2^C:

$$S \in \mathrm{Gen}(C,u), S \subseteq T \subseteq C \Longrightarrow T \in \mathrm{Gen}(C,u).$$

Proof. Monotonicity gives

$$C = C_2(S;u) \subseteq C_2(T;u) \subseteq C.$$

Hence equality holds. □

Thus minimal generating signatures determine all larger preserving signatures.

Definition 3.4. A generating signature $S$is inclusion-minimal when no proper subset of Sgenerates C.

The set of minimal generating signatures is

$$\mathrm{MinGen}(C,u) = \{S \in \mathrm{Gen}(C,u) : \forall T \subsetneq S,\ C_2(T;u) \neq C\}.$$

Singleton generators are automatically inclusion-minimal.

A cooperative pair is precisely an element of

$$\mathrm{MinGen}(C,u) \cap \mathrm{Gen}_2(C,u).$$

Hence the absence of cooperative pairs may be written

$$\mathrm{MinGen}(C,u) \cap \mathrm{Gen}_2(C,u) = \emptyset.$$

3.6 The Subcomponent Poset

Every subset $S \subseteq C$generates a subcomponent

$$C_S = C_2(S;u) \subseteq C.$$

Let

$$\mathrm{Sub}_u(C) = \{C_2(S;u) : S \subseteq C\}.$$

Ordered by inclusion,

$$(\mathrm{Sub}_u(C), \subseteq)$$

is a finite poset.

The least element is the projection-unary component

$$C_{\emptyset} = C_2(\emptyset; u).$$

The greatest element is C.

For subcomponents $D, E \in \mathrm{Sub}_u(C)$, define

$$D \vee E = C_2(D \cup E; u).$$

This is the least subcomponent containing both $D$and E.

The intersection $D \cap E$contains the projections and is closed under u. It is also closed under binary superposition by operations lying in the intersection. Therefore

$$D \cap E \in \mathrm{Sub}_u(C).$$

Thus

$$D \wedge E = D \cap E.$$

Proposition 3.4. The subcomponent poset $\mathrm{Sub}_u(C)$is a finite lattice with

$$D \wedge E = D \cap E$$

and

$$D \vee E = C_2(D \cup E; u).$$

Proof. The intersection is the greatest lower bound. Closure of the union supplies the least upper bound. Finiteness follows from Cbeing finite. □

A cooperative pair $\{g, h\}$has the lattice-theoretic form

$$C_{\{g\}} \vee C_{\{h\}} = C$$

while

$$C_{\{g\}} < C \text{and} C_{\{h\}} < C.$$

The Stage 2 cooperative-pair search is therefore a search for two proper principal subcomponents whose join is the whole component.

The absence of cooperative pairs means that no two proper singleton-generated subcomponents join to C in the tested examples.

3.7 Closure Rank

For $S \subseteq C$, let

$$r(S) =\mid C_2(S;u) \mid$$

be the generated-component cardinality.

Then

$$r(\emptyset) \leq r(S) \leq \mid C \mid.$$

The function r is monotone:

$$S \subseteq T \Longrightarrow r(S) \leq r(T).$$

A preserving signature satisfies

$$r(S) =\mid C \mid.$$

A singleton generator g satisfies

$$r(\{g\}) =\mid C \mid.$$

A cooperative pair satisfies

$$r(\{g\}) <\mid C \mid,$$
$$r(\{h\}) <\mid C \mid,$$

but

$$r(\{g,h\}) =\mid C \mid.$$

The incremental closure gain of adding $h$to $S$is

$$\delta_r(h \mid S) = r(S \cup \{h\}) - r(S).$$

This quantity measures newly accessible binary content, not reduced representation cost.

Primitive promotion inside an already preserving signature has

$$\delta_r(h \mid S) = 0.$$

Its value lies instead in complexity reduction.

This yields a sharp distinction between two forms of primitive addition:

$$\delta_r(h \mid S) > 0$$

means content expansion, whereas

$$\delta_r(h \mid S) = 0$$

with reduced term costs means conservative refactoring.

3.8 Generation and Refactoring

The same primitive addition may be evaluated along two independent axes.

The content gain is

$$\delta_r(h \mid S) = \mid C_2(S \cup \{h\}; u) \mid - \mid C_2(S; u) \mid.$$

The access gain, when both signatures are compared on a common component, is

$$\Delta_T(S, h) = T(S) - T(S \cup \{h\}).$$

This produces four conceptual cases.

δ_r	Δ_T	Interpretation
> 0	> 0	content expansion with improved access
> 0	$= 0$	content expansion without measured compression
$= 0$	> 0	conservative macro promotion
$= 0$	$= 0$	redundant primitive addition

The Stage 2 pair improvements lie in the third class.

They preserve the component:

$$\delta_r = 0,$$

while lowering total minimum size:

$$\Delta_T > 0.$$

A genuinely cooperative pair would be detected before preservation is reached. Starting from neither singleton, the joint addition would satisfy

$$r(\{g, h\}) =\mid C \mid$$

while

$$r(\{g\}), r(\{h\}) <\mid C \mid.$$

Cooperation concerns content formation. Promotion concerns access optimization after content is already present.

3.9 Relabeling Invariance

The representation of an operation table depends on the names assigned to carrier elements. The algebraic component should not.

Let $\rho: A \to A$be a permutation. For an n-ary operation f, define the relabeled operation

$$f^{\rho}(a_1, \dots, a_n) = \rho\,(f(\rho^{-1}(a_1), \dots, \rho^{-1}(a_n))).$$

For a unary u,

$$u^{\rho} = \rho \circ u \circ \rho^{-1}.$$

For a binary signature S,

$$S^{\rho} = \{b^{\rho}: b \in S\}.$$

Proposition 3.5. Relabeling commutes with binary closure:

$$C_2(S^\rho; u^\rho) = \{f^\rho : f \in C_2(S; u)\}.$$

Proof. The relabeled projections are the projections. Relabeling commutes with unary composition:

$$(u \circ f)^\rho = u^\rho \circ f^\rho.$$

It also commutes with binary superposition:

$$b(f, g)^\rho = b^\rho(f^\rho, g^\rho).$$

Induction over term construction yields the result. □

Corollary 3.2. Component cardinality is invariant under carrier relabeling:

$$| C_2(S^\rho; u^\rho) | = | C_2(S; u) |.$$

The minimum-size and minimum-height profiles are also invariant when primitive costs are transferred along the relabeling.

Proposition 3.6. If $f \mapsto f^\rho$ bijectively transfers a signature S to S^ρ, then

$$s_{S^\rho}(f^\rho) = s_S(f)$$

and

$$h_{S^\rho}(f^\rho) = h_S(f).$$

Proof. Relabeling transforms every S-term into an S^ρ-term with the same syntax tree. The inverse relabeling gives the reverse inequality. □

Thus exact complexity spectra are isomorphism invariants of pointed presentations.

3.10 Arity-Boundedness

The binary component records all operations induced by terms in two free variables. It need not determine the full term clone.

Let

$$\mathrm{Clo}(U, S) = \bigcup_{n \geq 1} \mathrm{Cl}_n\,(U, S)$$

be the complete finitary clone generated by the signature.

Equality

$$C_2(S;u) = C_2(T;u)$$

does not automatically imply

$$\mathrm{Clo}(\{u\},S) = \mathrm{Clo}(\{u\},T).$$

The release therefore uses the language:

binary-component preserving,

not unrestricted term-equivalent, unless higher-arity equality is separately established.

Definition 3.5. A refactoring is arity-n preserving when

$$\mathrm{Cl}_n(U,S) = \mathrm{Cl}_n(U,T).$$

It is fully clone-preserving when equality holds for every $n \geq 1$.

The present computations establish arity-two preservation:

$$\mathrm{Cl}_2(\{u\},S) = \mathrm{Cl}_2(\{u\},T).$$

This restriction is both methodological and computational. For fixed finite A, every arity-n operation space is finite, but its size grows as

$$q^{q^n}.$$

Even small increases in $n$create substantial expansion.

For $q = 3$,

$$|\ \mathcal{O}_A^{(2)}\ | = 3^9 = 19{,}683,$$

whereas

$$| \mathcal{O}_A^{(3)} | = 3^{27} = 7{,}625{,}597{,}484{,}987.$$

Binary analysis is therefore the first exact tractable layer, not the final theory.

3.11 Finite Presentation Space

For a component C of cardinality N, the number of unordered signatures of width k is

$$\binom{N}{k}.$$

The number of signatures of width at most k is

$$\sum_{j=1}^{k} \binom{N}{j}.$$

For width two,

$$| \{S \subseteq C : | S | = 2\} | = \binom{N}{2}.$$

The principal components therefore produce:

$$\binom{16}{2} = 120,$$
$$\binom{24}{2} = 276,$$

and

$$\binom{82}{2} = 3{,}321$$

unordered pairs.

These are the exact pair counts used in the exhaustive signature-refactoring census.

The width-two search is finite and complete relative to each fixed component. Every unordered pair of distinct binary functions is tested.

The result is not a sample of signatures. It is the complete width-two presentation space.

3.12 Component-Preserving Signature Space

For a pointed component (C, u), define

$$\mathcal{R}_k(C,u) = \{S \subseteq C : | \, S \, | = k,\ C_2(S;u) = C\}.$$

The complete bounded refactoring space is

$$\mathcal{R}_{\leq k}(C,u) = \bigcup_{j=1}^{k} \mathcal{R}_j\,(C,u).$$

For $k = 2$,

$$\mathcal{R}_{\leq 2}(C,u) = \mathcal{R}_1(C,u) \cup \mathcal{R}_2(C,u).$$

The nonpreserving region is

$$\mathcal{N}_k(C,u) = \{S \subseteq C : | \, S \, | = k,\ C_2(S;u) \neq C\}.$$

Thus

$$\binom{C}{k} = \mathcal{R}_k(C,u) \sqcup \mathcal{N}_k(C,u).$$

The preserving density is

$$p_k(C,u) = \frac{| \, \mathcal{R}_k(C,u) \, |}{\binom{|C|}{k}}.$$

For width two, the computed densities are

$$p_2(C_B,u_B) = \frac{92}{120},$$
$$p_2(C_W,u_W) = \frac{156}{276},$$

and

$$p_2(C_S, u_S) = \frac{620}{3321}.$$

These densities quantify how common content-preserving pair signatures are within the full pair space.

They do not quantify efficiency. That requires exact cost profiles.

3.13 Structural Separation

The finite pointed-component framework separates four logically distinct questions.

First:

Does S generate C?

This is the preservation problem.

Second:

How expensive is each $f \in C$ relative to S?

This is the representation problem.

Third:

Does S contain a smaller generating subset?

This is the minimality problem.

Fourth:

How should primitive width be traded against derivational cost?

This is the optimization problem.

A signature may preserve C but be inefficient. It may be efficient but nonminimal. It may minimize total cost while failing to minimize maximum height. It may dominate under zero primitive penalty and lose under positive vocabulary cost.

No single completeness predicate resolves these distinctions.

The pointed component supplies the invariant content. The preserving-signature space supplies the admissible presentations. Exact cost functions supply the geometry.

The subsequent section constructs those cost functions independently for operation-node size and tree height.

4. Exact Minimum Size and Minimum Height

4.1 Two Independent Complexity Measures

Let (C, u)be a finite pointed binary term-function component and let $S \subseteq C$be a preserving signature.

Every term $t$in variables x, y, unary primitive u, and binary primitives Shas both an operation-node size and a tree height.

The size function is defined recursively by

$$s(x) = s(y) = 0,$$
$$s(u(t)) = s(t) + 1,$$

and

$$s(b(t_1, t_2)) = s(t_1) + s(t_2) + 1 (b \in S).$$

The height function is defined independently by

$$h(x) = h(y) = 0,$$
$$h(u(t)) = h(t) + 1,$$

and

$$h(b(t_1, t_2)) = \max\{h(t_1), h(t_2)\} + 1.$$

For $f \in C$, define

$$s_S(f) = \min\{s(t) : t^A = f\}$$

and

$$h_S(f) = \min\{h(t) : t^A = f\}.$$

The two minima are taken independently. A term minimizing size need not minimize height, and a term minimizing height need not minimize size.

Definition 4.1. The exact size profile of $S$is

$$\mathbf{s}_S = (s_S(f))_{f \in C}.$$

The exact height profile is

$$\mathbf{h}_S = (h_S(f))_{f \in C}.$$

Because $C$is finite and $S$is preserving, both profiles are finite-valued.

4.2 Global Profile Statistics

Several scalar invariants are extracted from the profiles.

The total minimum-size cost is

$$T(S) = \sum_{f \in C} s_S(f).$$

The saturation size is

$$S_{\max}(S) = \max_{f \in C} s_S(f).$$

The total minimum-height cost is

$$U(S) = \sum_{f \in C} h_S(f).$$

The saturation height is

$$H_{\max}(S) = \max_{f \in C} h_S(f).$$

The size histogram is

$$N_k^s(S) = |\{f \in C : s_S(f) = k\}|,$$

and the height histogram is

$$N_k^h(S) = |\{f \in C : h_S(f) = k\}|.$$

These satisfy

$$\sum_{k\geq 0} N_k^s(S) = |C|,$$
$$\sum_{k\geq 0} k\, N_k^s(S) = T(S),$$
$$\sum_{k\geq 0} N_k^h(S) = |C|,$$

and

$$\sum_{k\geq 0} k\, N_k^h(S) = U(S).$$

The saturation statistics identify the most remote functions in the presentation. The totals describe the global cost of reaching the entire component.

4.3 Size and Height Are Not Interchangeable

For every term t,

$$h(t) \leq s(t).$$

This follows because every root-to-leaf path contains no more operation nodes than the full syntax tree.

Consequently,

$$h_S(f) \leq s_S(f)$$

for every $f \in C$.

The converse does not hold with equality in general.

A balanced binary tree may have large size and small height. A unary chain has equal size and height. Repeated use of two large subterms can increase size additively while height increases only by one beyond the larger branch.

Proposition 4.1. For every preserving signature $S$and every $f \in C$,

$$h_S(f) \leq s_S(f).$$

Proof. For every term t,

$$h(t) \leq s(t)$$

by induction on term formation. Taking the minimum over all representations of $f$gives

$$h_S(f) \leq h(t_s) \leq s(t_s) = s_S(f),$$

where $t_s$is a minimum-size representation. □

The inequality does not permit height to be reconstructed from size.

Proposition 4.2. Equality of exact size profiles does not imply equality of exact height profiles.

Proof. Size aggregates all operation nodes, whereas height records only the longest root-to-leaf path. Two primitive systems may induce representations with equal minimum node counts but different optimal nestings. Therefore the size profile does not determine the height profile. □

The release computes the two profiles independently. Earlier identification of saturation size with saturation height would therefore be invalid.

4.4 Extensional Dynamic Programming

The finite component $C$is represented extensionally. Every $f \in C$is identified by its complete table on A^2.

The minimum-size problem may therefore be formulated as finite dynamic programming over function tables.

Initialize

$$d_s(\pi_1) = d_s(\pi_2) = 0.$$

For every binary primitive $b \in S$,

$$d_s(b) \leq 1.$$

Unary extension gives the relaxation rule

$$d_s(u \circ f) \leq d_s(f) + 1.$$

Binary superposition gives

$$d_s(b(f, g)) \leq d_s(f) + d_s(g) + 1.$$

The exact minimum-size profile is the least assignment satisfying all such inequalities and the base conditions.

Similarly, for height,

$$\begin{aligned} &d_h(\pi_1) = d_h(\pi_2) = 0, \\ &d_h(b) \leq 1(b \in S), \\ &d_h(u \circ f) \leq d_h(f) + 1, \end{aligned}$$

and

$$d_h(b(f, g)) \leq \max\{d_h(f), d_h(g)\} + 1.$$

The exact minimum-height profile is the least assignment satisfying the corresponding height inequalities.

Definition 4.2. A size-feasible labeling is a map

$$d: C \to \mathbb{N}$$

satisfying the size base conditions and all size relaxation inequalities.

A height-feasible labeling is defined analogously.

The minimum profiles are pointwise least feasible labelings.

Proposition 4.3. The maps $s_S$and $h_S$are feasible labelings.

Proof. Each defining inequality follows by composing minimum representations of the input functions and observing that the resulting term is a representation of the output. □

Proposition 4.4. If $d$is any size-feasible labeling, then

$$d(f) \leq s(t)$$

for every term $t$representing f.

Proof. Proceed by induction on the syntax of t. The variable and primitive cases follow from the base conditions. Unary and binary cases follow from the feasibility inequalities. □

Hence

$$d(f) \leq s_S(f).$$

For the greatest feasible lower bound obtained by relaxation to closure, equality holds.

4.5 Weighted Derivation Hypergraph

The exact computation may be represented as a finite directed hypergraph.

The vertex set is C.

Unary constructions give directed edges

$$f \longrightarrow u \circ f.$$

Binary constructions give directed hyperedges

$$(f, g) \longrightarrow b(f, g)(b \in S).$$

For size, the cost of a unary derivation is

$$c_s(f \to u \circ f) = d_s(f) + 1,$$

and the cost of a binary derivation is

$$c_s((f, g) \to b(f, g)) = d_s(f) + d_s(g) + 1.$$

For height,

$$c_h(f \to u \circ f) = d_h(f) + 1$$

and

$$c_h((f, g) \to b(f, g)) = \max\{d_h(f), d_h(g)\} + 1.$$

The derivation hypergraph is finite because C is finite.

Every nontrivial term corresponds to a rooted derivation tree in this hypergraph. Conversely, every derivation tree determines a term.

Thus minimum term size and minimum term height are shortest-derivation problems under two different cost algebras.

4.6 Monotone Finalization

All constructors strictly increase both size and height relative to each immediate subterm:

$$s(u(t)) > s(t),$$
$$s(b(t_1, t_2)) > s(t_1),$$
$$s(b(t_1, t_2)) > s(t_2),$$

and

$$h(u(t)) > h(t),$$
$$h(b(t_1, t_2)) > h(t_1),$$
$$h(b(t_1, t_2)) > h(t_2).$$

Therefore a function cannot acquire a cheaper representation through a derivation whose constituent functions have not yet been reached at lower cost.

This permits a priority-queue algorithm analogous to Dijkstra finalization, generalized to binary hyperedges.

For size, the queue key is the provisional value $d_s(f)$. When f is removed with the least current key, every unary construction from f is relaxed. Every binary construction involving f and every already discovered g is also relaxed.

For height, the same process is used with the height relaxation rule.

Algorithm 4.1. Exact minimum-size profile

Input:

$$(A, u, S, C).$$

Initialize:

$$d_s(f) = \infty (f \in C).$$

Set:

$$d_s(\pi_1) = d_s(\pi_2) = 0,$$

and

$$d_s(b) = \min\{d_s(b),1\} (b \in S).$$

Insert all finite entries into a minimum-priority queue.

Repeatedly remove an unsettled function f with least key.

Relax:

$$u \circ f$$

with candidate cost

$$d_s(f) + 1.$$

For every g whose current distance is finite and every $b \in S$, relax

$$b(f, g)$$

with candidate cost

$$d_s(f) + d_s(g) + 1$$

and

$$b(g, f)$$

with the same candidate cost.

Continue until the queue is empty.

Output:

$$d_s = s_S.$$

Algorithm 4.2. Exact minimum-height profile

Use the same initialization, replacing binary candidate size by

$$\max\{d_h(f), d_h(g)\} + 1.$$

Output:

$$d_h = h_S.$$

4.7 Correctness of Minimum-Size Computation

Theorem 4.1. Algorithm 4.1 returns the exact minimum operation-node size $s_S(f)$for every $f \in C$.

Proof. Every relaxed value is the size of an explicit term constructed from previously realized terms. Therefore

$$d_s(f) \geq s_S(f)$$

cannot occur at termination through an invalid underestimate; every finite label is realizable.

Assume for contradiction that some function $f$is finalized with

$$d_s(f) > s_S(f).$$

Choose such an $f$with minimum true size, and let $t$be a minimum-size term representing f.

If $t$is a variable or primitive, the initialization gives the exact value.

If

$$t = u(r),$$

then the function $g = r^A$satisfies

$$s_S(g) < s_S(f).$$

By minimality of f, $g$is assigned its exact size before $f$is finalized. Relaxation of the unary construction yields

$$d_s(f) \leq s_S(g) + 1 = s_S(f),$$

contradicting the assumed strict inequality.

If

$$t = b(r_1, r_2),$$

let

$$g_1 = r_1^A, g_2 = r_2^A.$$

Then

$$s_S(g_1) < s_S(f)$$

and

$$s_S(g_2) < s_S(f).$$

Both receive exact values before f is finalized. Binary relaxation yields

$$d_s(f) \leq s_S(g_1) + s_S(g_2) + 1 = s_S(f),$$

again a contradiction.

Therefore every finalized value is exact. □

4.8 Correctness of Minimum-Height Computation

Theorem 4.2. Algorithm 4.2 returns the exact minimum tree height $h_S(f)$for every $f \in C$.

Proof. Every provisional finite height is realized by an explicit term, so the algorithm never assigns a value below the true minimum.

Assume that f is the first function finalized with

$$d_h(f) > h_S(f),$$

and let $t$be a minimum-height term representing f.

The variable and primitive cases are exact at initialization.

If

$$t = u(r),$$

then

$$h(r) < h(t).$$

The function $g = r^A$has already received its exact height, and unary relaxation gives

$$d_h(f) \leq h_S(g) + 1 = h_S(f).$$

If

$$t = b(r_1, r_2),$$

then both subterms have height strictly smaller than $h(t)$. Their induced functions therefore receive exact labels before f. Binary relaxation gives

$$d_h(f) \leq \max\{h_S(g_1), h_S(g_2)\} + 1 = h_S(f).$$

Both cases contradict the assumption. □

The proof depends on strict growth of height along every constructor.

4.9 Witness Terms

The computation may store, together with each exact value, a witness term.

For size, if the best relaxation for $f$is unary, store

$$w_s(f) = u(w_s(g)).$$

If it is binary, store

$$w_s(f) = b(w_s(g_1), w_s(g_2)).$$

The resulting witness satisfies

$$w_s(f)^A = f$$

and

$$s(w_s(f)) = s_S(f).$$

Similarly, a height witness $w_h(f)$satisfies

$$w_h(f)^A = f$$

and

$$h(w_h(f)) = h_S(f).$$

The two witnesses may differ.

Definition 4.3. A certified minimum-size record for $f$is a tuple

$$(\mathrm{tab}(f), s_S(f), w_s(f))$$

such that direct evaluation verifies

$$w_s(f)^A = f$$

and direct syntax counting verifies

$$s(w_s(f)) = s_S(f).$$

A minimum-height record is defined analogously.

A witness establishes the upper bound. Exactness of the algorithm establishes the matching lower bound.

4.10 Independent Oracles

An implementation error may reproduce itself across related routines. The release therefore separates several forms of verification.

The principal profile computation uses priority-queue closure.

Independent checks may use:

layered breadth-first enumeration,
fixed-point relaxation,

or

direct saturation by increasing cost.

For minimum size, define layers

$$L_k^S = \{f \in C : s_S(f) \leq k\}.$$

Then

$$L_0^S = \{\pi_1, \pi_2\}.$$

Functions enter L_k^S when they admit either:

$$u \circ f \text{ with } f \in L_{k-1}^S,$$

or

$$b(f, g)$$

with exact constituent costs i, j satisfying

$$i + j + 1 \leq k.$$

For height, define

$$L_k^h = \{f \in C : h_S(f) \leq k\}.$$

Then

$$L_{k+1}^h$$

is obtained by closing L_k^h under u and under binary superposition of members of L_k^h.

Because height uses a maximum recurrence, height layers have the particularly simple form

$$L_{k+1}^h = L_k^h \cup u(L_k^h) \cup \bigcup_{b \in S} b(L_k^h, L_k^h).$$

Proposition 4.5. The least $k$for which $f \in L_k^h$is $h_S(f)$.

Proof. Induction on $k$shows that L_k^hcontains exactly the functions representable by terms of height at most k. □

This layered oracle is structurally different from the priority-queue implementation and therefore provides an independent verification route.

4.11 Saturation

For size, define the size saturation threshold

$$\sigma_s(S) = \min\{k : \{f : s_S(f) \leq k\} = C\}.$$

Then

$$\sigma_s(S) = S_{\max}(S).$$

For height,

$$\sigma_h(S) = \min\{k : \{f : h_S(f) \leq k\} = C\}.$$

Then

$$\sigma_h(S) = H_{\max}(S).$$

The terms saturation size and saturation height refer to different quantities.

The former is the largest minimum number of operation nodes required by any function. The latter is the largest minimum nesting depth required by any function.

A component may saturate at size $7$and height 4. Reporting the pair as $7/7$, or deriving one coordinate from the other, would conflate distinct invariants.

4.12 Diameter Interpretation

The primitive presentation induces a directed derivation geometry on C.

The projections are roots of cost zero. Primitive binary functions have cost at most one. Every other function lies at a finite derivational distance.

The saturation size

$$S_{\max}(S)$$

is the maximum additive derivation distance from the primitive base under the size cost.

The saturation height

$$H_{\max}(S)$$

is the maximum parallel-depth distance under the height cost.

These are two different diameters of the same extensional component.

Size models sequential symbolic work:

total number of operation occurrences.

Height models idealized parallel depth:

longest dependency chain.

The same signature may be favorable under one interpretation and unfavorable under the other.

4.13 Dominance Between Signatures

Let $S, T \in \mathcal{R}(C, u)$.

Definition 4.4. Signature Ssize-dominates $T$when

$$s_S(f) \leq s_T(f)$$

for every $f \in C$, with strict inequality for at least one f.

Write

$$S \prec_s T.$$

Height dominance is defined by

$$h_S(f) \leq h_T(f)$$

for every f, with strict inequality somewhere.

Joint dominance requires both.

If $S \subseteq T$, then monotonicity gives

$$s_T(f) \leq s_S(f)$$

and

$$h_T(f) \leq h_S(f).$$

Thus primitive promotion never worsens either profile.

For unrelated signatures, neither need dominate the other. One may make some functions cheaper and others more expensive.

Definition 4.5. The Pareto set of a preserving-signature family $\mathcal{F}$ is

$$\text{Pareto}(\mathcal{F}) = \{S \in \mathcal{F} : \nexists T \in \mathcal{F} \text{ jointly dominating } S\}.$$

The Pareto set may contain several signatures even when one uniquely minimizes total size.

4.14 Primitive Width and Trivial Compression

Every $b \in S$ has

$$s_S(b) \leq 1$$

and

$$h_S(b) \leq 1.$$

Adding primitives therefore lowers some costs by definition. Comparisons across widths must account for this built-in advantage.

Without a primitive-width penalty, the full signature

$$S = C$$

trivially gives

$$s_C(f) \leq 1$$

for every nonprojection $f \in C$.

Hence unconstrained minimization of $T(S)$over all preserving signatures is not informative. The optimum would be achieved by declaring most or all functions primitive.

The meaningful problem requires either:

$$| S | \leq k,$$

or a width penalty such as

$$J_\lambda(S) = T(S) + \lambda(| S | -1).$$

Width-constrained optimization asks:

$$\min\{T(S): S \in \mathcal{R}_k(C, u)\}.$$

Penalized optimization asks:

$$\min\{J_\lambda(S): S \in \mathcal{R}(C, u)\}.$$

The Stage 2 exhaustive comparison uses the width constraint

$$| S | \leq 2$$

and subsequently evaluates a one-parameter width penalty.

4.15 Exact Promotion Gain

Let Sbe preserving and let $m \in C \setminus S$.

Set

$$S^+ = S \cup \{m\}.$$

For every $f \in C$,

$$s_{S^+}(f) \leq s_S(f).$$

Define the pointwise size gain

$$\delta^S_{S,m}(f) = s_S(f) - s_{S^+}(f).$$

Then

$$\delta^{S}_{S,m}(f) \geq 0.$$

The total gain is

$$\Delta_T(S,m) = \sum_{f \in C} \delta^{S}_{S,m}(f).$$

Similarly,

$$\delta^{h}_{S,m}(f) = h_S(f) - h_{S^+}(f)$$

and

$$\Delta_U(S,m) = \sum_{f \in C} \delta^{h}_{S,m}(f).$$

Promotion gain may be decomposed as

$$\Delta_T(S,m) = (s_S(m) - 1) + \sum_{f \neq m} \delta^{S}_{S,m}(f).$$

The first term is direct compression of the promoted function. The second is propagated compression through reuse of the macro.

Definition 4.6. The propagation gain is

$$P_T(S,m) = \sum_{f \neq m} \delta^{S}_{S,m}(f).$$

Thus

$$\Delta_T(S,m) = s_S(m) - 1 + P_T(S,m).$$

A promoted operation is globally useful when $P_T(S,m)$is large, not merely when the operation itself was expensive.

4.16 Exactness Under Ties

Several different terms may realize the same minimum size or height.

The profile depends only on the minimum value, not on uniqueness of the witness.

To make computational records deterministic, ties are resolved by a frozen canonical policy. A typical ordering may compare:

1. cost;
2. constructor class;
3. primitive code;
4. left witness code;
5. right witness code;
6. serialized syntax.

The tie-breaking rule does not change

$$s_S(f)$$

or

$$h_S(f).$$

It changes only which minimum witness is stored.

Proposition 4.6. Any deterministic tie-breaking policy among equal-cost witnesses preserves the exact complexity profiles.

Proof. All tied witnesses realize the same function at the same minimum cost. Selecting one does not alter the minimum value. □

This distinction separates mathematical invariants from archival representation choices.

4.17 Complexity Bounds

Let

$$N = \mid C \mid$$

and

$$k = |\ S\ |.$$

A direct full relaxation sweep examines at most

$$N$$

unary constructions and

$$kN^2$$

binary constructions.

A naive fixed-point implementation therefore has per-sweep cost

$$O(kN^2)$$

table operations.

The number of strict distance improvements is finite. With a priority queue, implementation cost depends on the number of successful relaxations and the strategy used to combine newly finalized functions with existing ones.

The extensional cost of one binary superposition is

$$O(|\ A\ |^2)$$

because the resulting binary function table has q^2entries.

A direct upper-bound form is therefore

$$O(kN^2q^2)$$

per complete combination pass.

For the principal components,

$$N \in \{16,24,82\}.$$

Exact singleton and pair profiles are therefore tractable even when every candidate signature is evaluated exhaustively.

The pair landscape requires

$$\binom{N}{2}$$

separate closure-and-profile computations. The Strong Kleene case is consequently the dominant run:

$$\binom{82}{2} = 3321.$$

4.18 Verification Conditions

A reported profile is accepted only when the following conditions hold.

First, every function in the claimed component is reached:

$$\{f : d_s(f) < \infty\} = C.$$

Second, every stored witness evaluates to the recorded function.

Third, the syntax-derived cost of every witness equals its recorded cost.

Fourth, no relaxation inequality is violated.

For size:

$$s_S(u \circ f) \leq s_S(f) + 1,$$
$$s_S(b(f, g)) \leq s_S(f) + s_S(g) + 1.$$

For height:

$$h_S(u \circ f) \leq h_S(f) + 1,$$
$$h_S(b(f, g)) \leq \max\{h_S(f), h_S(g)\} + 1.$$

Fifth, independent recomputation reproduces the same normalized profile.

Definition 4.7. A normalized profile record removes incidental witness serialization and retains:

$$\mathrm{tab}(f), s_S(f), h_S(f)$$

for every $f \in C$, together with aggregate statistics and signature identifiers.

Normalized comparison verifies scientific content independently of harmless changes in witness formatting.

4.19 Corrected Size-Height Semantics

The exact-signature release distinguishes four quantities:

$$T(S),$$
$$S_{\max}(S),$$
$$U(S),$$

and

$$H_{\max}(S).$$

The notation is chosen to prevent the maximum minimum node count from being described as a height.

The correct paired summary is

$$S_{\max}(S)/H_{\max}(S).$$

For example,

$$3/2$$

means that every function has a representation using at most three operation nodes, while every function has a representation of height at most two. It does not mean that one common witness simultaneously realizes both minima.

Similarly,

$$5/3$$

reports separate optimal radii in the size and height geometries.

This correction is mathematically substantive. It changes the interpretation of the formation profile from a single scalar depth to a pair of independent complexity radii.

4.20 Formation Profiles

For a preserving signature S, define the joint formation profile

$$\Psi(S) = (|\ S\ |, |\ C\ |, T(S), U(S), S_{\max}(S), H_{\max}(S), \mathbf{s}_S, \mathbf{h}_S).$$

The profile records how the whole component becomes accessible from the selected primitive vocabulary.

Two signatures may satisfy

$$C_2(S; u) = C_2(T; u)$$

while

$$\Psi(S) \neq \Psi(T).$$

This is the central finite phenomenon studied in the remaining cycles.

The component supplies the invariant mathematical content.

The formation profile supplies the presentation-relative geometry.

Exact signature refactoring is the study of the map

$$S \longmapsto \Psi(S)$$

over the preserving-signature space.

5. Presentation-Dependent Formation Profiles

5.1 Fixed Content, Variable Access

Let (C, u)be a finite pointed binary term-function component. Let

$$g, h \in C$$

be singleton generators satisfying

$$C_2(\{g\}; u) = C$$

and

$$C_2(\{h\}; u) = C.$$

The signatures

$$\Sigma_g = \{u, g\}$$

and

$$\Sigma_h = \{u, h\}$$

have the same binary extensional content. Nevertheless, they need not induce the same minimum-size or minimum-height functions:

$$\mathbf{s}_{\{g\}} \neq \mathbf{s}_{\{h\}},$$
$$\mathbf{h}_{\{g\}} \neq \mathbf{h}_{\{h\}}.$$

The distinction is not caused by a change in the carrier, term-function component, or admissible output operations. It is caused solely by the choice of which binary operation is assigned primitive cost one.

For each singleton generator g, define its formation profile by

$$\Psi(g) = \big(T(g), U(g), S_{\max}(g), H_{\max}(g), \mathbf{s}_g, \mathbf{h}_g\big),$$

where

$$T(g) = \sum_{f \in C} s_g\,(f),$$
$$U(g) = \sum_{f \in C} h_g\,(f),$$
$$S_{\max}(g) = \max_{f \in C} s_g(f),$$

and

$$H_{\max}(g) = \max_{f \in C} h_g(f).$$

The map

$$g \longmapsto \Psi(g)$$

assigns a distinct generative geometry to each primitive presentation.

A component may therefore be extensionally fixed while supporting a nontrivial spectrum of formation profiles.

Definition 5.1. The singleton formation spectrum of $(C, u)$is

$$\mathfrak{F}_1(C,u) = \{\Psi(g) : g \in G(C,u)\},$$

where

$$G(C,u) = \{g \in C : C_2(\{g\}; u) = C\}.$$

If

$$|\, \mathfrak{F}_1(C,u) \,| > 1,$$

then the component is presentation-sensitive under the declared complexity measures.

The Stage 2 calculations establish presentation sensitivity in every principal family examined.

5.2 Involutive Polarity Transformations

The Boolean and Kleene examples contain a distinguished unary involution u:

$$u^2 = \mathrm{id}_A.$$

For a binary operation b, one may alter output polarity, input polarities, or argument order while retaining interdefinability over u.

Let

$$\varepsilon_0, \varepsilon_1, \varepsilon_2 \in \{0,1\},$$

and let

$$\sigma \in S_2.$$

Write

$$u^0 = \mathrm{id}_A, u^1 = u.$$

Define

$$b^{(\varepsilon_0;\varepsilon_1,\varepsilon_2;\sigma)}(x_1,x_2)=u^{\varepsilon_0}\Big(b\big(u^{\varepsilon_1}(x_{\sigma(1)}),u^{\varepsilon_2}(x_{\sigma(2)})\big)\Big).$$

The transformations are generated by:

output polarity reversal,
first-input polarity reversal,
second-input polarity reversal,

and

argument exchange.

The resulting finite set is the polarity-permutation orbit

$$\mathrm{Orb}_u(b)=\big\{b^{(\varepsilon_0;\varepsilon_1,\varepsilon_2;\sigma)}\big\}.$$

Orbit cardinality may be smaller than sixteen because different transformations can produce the same operation.

Proposition 5.1. Every operation

$$b'\in\mathrm{Orb}_u(b)$$

is term-definable from $u$and b.

Proof. By definition, $b'$is represented by the term

$$u^{\varepsilon_0}\Big(b\big(u^{\varepsilon_1}(x_{\sigma(1)}),u^{\varepsilon_2}(x_{\sigma(2)})\big)\Big).$$

Hence

$$b'\in C_2(\{b\};u).$$

□

Each polarity-permutation transformation is invertible because $u$and $\sigma$are invertible.

Corollary 5.1.

$$C_2(\{b'\}; u) = C_2(\{b\}; u)$$

for every

$$b' \in \mathrm{Orb}_u(b).$$

Thus a polarity orbit supplies a natural family of binary-component-preserving primitive substitutions.

The operations in the orbit are equivalent with respect to generated binary content. They need not be equivalent with respect to exact formation cost.

5.3 Cost of Polarity Translation

The direct representation of

$$b' = b^{(\varepsilon_0;\varepsilon_1,\varepsilon_2;\sigma)}$$

in the signature $\{u, b\}$has size

$$1 + \varepsilon_0 + \varepsilon_1 + \varepsilon_2.$$

Its height is at most

$$1 + \varepsilon_0 + \max\{\varepsilon_1, \varepsilon_2\}.$$

The argument permutation contributes no operation node because variables may be exchanged syntactically at zero cost.

Therefore

$$s_{\{b\}}(b') \leq 1 + \varepsilon_0 + \varepsilon_1 + \varepsilon_2$$

and

$$h_{\{b\}}(b') \leq 1 + \varepsilon_0 + \max\{\varepsilon_1, \varepsilon_2\}.$$

When $b'$is itself selected as primitive,

$$s_{\{b'\}}(b') = 1, h_{\{b'\}}(b') = 1.$$

The translation overhead disappears for b'but may reappear elsewhere in the component.

This explains why polarity-related primitives can generate identical content while producing unequal global profiles.

Proposition 5.2. Polarity-orbit membership implies binary-component equality but does not imply equality of exact size or height profiles.

Proof. Component equality follows from Corollary 5.1. Profile equality would require every function to admit representations of equal minimum cost in both signatures. However, b'has primitive size one in its own signature and may require multiple unary decorations when represented from b. Conversely, $b$may require corresponding decorations from b'. These local asymmetries can propagate through larger terms. Therefore no general equality of profiles follows. □

The polarity orbit is consequently a controlled test family for presentation dependence.

The algebraic content is fixed by exact interdefinability. The primitive access point is changed.

5.4 Polarity-Orbit Formation Spectrum

For a generator b, define its orbit spectrum by

$$\mathfrak{F}_{\mathrm{orb}}(b; u) = \{\Psi(b') : b' \in \mathrm{Orb}_u(b)\}.$$

The orbit is formation-uniform when

$$| \mathfrak{F}_{\mathrm{orb}}(b; u) | = 1.$$

It is formation-nonuniform when

$$| \mathfrak{F}_{\mathrm{orb}}(b; u) | > 1.$$

Define the orbit size spread

$$\omega_s(b) = \max_{b' \in \mathrm{Orb}_u(b)} S_{\max}(b') - \min_{b' \in \mathrm{Orb}_u(b)} S_{\max}(b'),$$

and the orbit height spread

$$\omega_h(b) = \max_{b' \in \mathrm{Orb}_u(b)} H_{\max}(b') - \min_{b' \in \mathrm{Orb}_u(b)} H_{\max}(b').$$

Similarly, define the total-cost spreads

$$\omega_T(b) = \max_{b'} T(b') - \min_{b'} T(b')$$

and

$$\omega_U(b) = \max_{b'} U(b') - \min_{b'} U(b').$$

A nonzero spread establishes presentation dependence within a tightly controlled interdefinability class.

5.5 Cycle 7 Exact Profile Calculation

Cycle 7 computed joint minimum-size and minimum-height profiles for binary primitives in four finite families:

Boolean meet-negation,
Weak Kleene,
Strong Kleene,

and

the three-element BCK-related union component.

For every tested primitive g, the computation determined:

$C_2(\{g\}; u)$,
$s_g(f) (f \in C_2)$,
$h_g(f) (f \in C_2)$,
$T(g)$,
$U(g)$,
$S_{\max}(g)$,

and

$$H_{\max}(g).$$

Only primitives regenerating the declared reference component were included in the preserving-signature spectrum.

The principal saturation ranges were:

Family	$\mid C_2 \mid$	$S_{\max}$ range	$H_{\max}$ range
Boolean meet-negation	16	5–7	3–4
Weak Kleene	24	5–7	3–4
Strong Kleene	82	6–10	4–5
BCK-related union	1152	10–13	5

These ranges are not ranges over different carriers or different generated components. Within each row, the component is held fixed while the primitive presentation varies.

Theorem 5.1. Each Cycle 7 reference component admits preserving singleton presentations with unequal exact formation profiles.

In particular, the saturation-size spreads are

$$2,2,4,3,$$

respectively.

The saturation-height spreads are

$$1,1,1,0.$$

The theorem establishes presentation dependence for exact finite term complexity.

5.6 Boolean Formation Profiles

Let

$$A_B = \{0,1\},$$

let $u_B$be Boolean complementation, and let $C_B$be the binary term-function component generated from meet and complementation.

The component has

$$\mid C_B \mid = 16.$$

Thus every Boolean binary function belongs to the component.

Different singleton generators over the same fixed complementation produce saturation sizes ranging from

$$5$$

to

$$7$$

and saturation heights ranging from

$$3$$

to

$$4.$$

Hence there exist singleton generators $g, h \in C_B$ such that

$$C_2(\{g\}; u_B) = C_2(\{h\}; u_B) = C_B,$$

but

$$S_{\max}(g) < S_{\max}(h).$$

There also exist preserving primitive choices with different maximum minimum heights.

The component is therefore not associated with a unique exact access radius.

Corollary 5.2. Functional completeness of a Boolean binary primitive relative to complementation does not determine its exact global term-size or tree-height efficiency.

The result is elementary in setting but methodologically important. Even in the complete sixteen-function Boolean component, primitive choice changes the exact cost of reconstructing the component.

The content statement

$$C_2 = C_B$$

is too coarse to determine the formation geometry.

5.7 Weak Kleene Formation Profiles

Let

$$A_W = \{0, \perp, 1\},$$

where $\perp$is an indeterminate value, and let $u_W$be the Weak Kleene involution satisfying

$$u_W(0) = 1, u_W(1) = 0, u_W(\perp) = \perp.$$

Let $C_W$be the binary term-function component used in the release.

Its cardinality is

$$| C_W | = 24.$$

The preserving singleton presentations have saturation sizes in the range

$$5 \leq S_{\max} \leq 7$$

and saturation heights in the range

$$3 \leq H_{\max} \leq 4.$$

The numerical radii coincide with the Boolean ranges, but the underlying components differ:

$$| C_B | = 16, | C_W | = 24.$$

Therefore equality of saturation ranges does not imply equality of formation spectra.

The full profiles distinguish:

which functions occupy each cost layer,
how many functions appear at each layer,

and

which primitive minimizes total rather than maximum cost.

A scalar radius is only a coarse projection of the complete profile.

Define the cumulative size-growth function

$$F_g^S(k) = |\{f \in C_W : s_g(f) \leq k\}|.$$

Similarly,

$$F_g^h(k) = |\{f \in C_W : h_g(f) \leq k\}|.$$

Two signatures may share the same saturation size while differing at intermediate layers:

$$F_g^S(k) \neq F_h^S(k)$$

for some

$$k < S_{\max}.$$

Thus the formation spectrum records more than the final saturation threshold.

5.8 Strong Kleene Formation Profiles

Let

$$A_S = \{0, \perp, 1\}$$

with the same fixed-point involution

$$u_S(0) = 1, u_S(1) = 0, u_S(\perp) = \perp,$$

but with the Strong Kleene binary semantics used in the release.

The generated binary component has

$$|C_S| = 82.$$

Its singleton-generator formation spectrum is substantially broader in size than the Boolean and Weak Kleene spectra:

$$6 \leq S_{\max} \leq 10.$$

The height range is

$$4 \leq H_{\max} \leq 5.$$

Thus the size-radius spread is

$$\omega_s = 4,$$

while the height-radius spread is only

$$\omega_h = 1.$$

This demonstrates that presentation changes may affect total symbolic work more strongly than parallel nesting depth.

A primitive with saturation size ten may still have saturation height five. Another primitive may reduce the maximum node count to six while reducing height by only one.

The two complexity geometries therefore respond differently to primitive substitution.

Proposition 5.3. In the Strong Kleene component, variation in minimum operation-node size is not explained by an equal variation in minimum tree height.

Proof. The saturation-size range has width four, while the saturation-height range has width one. Hence changes in maximum minimum size cannot be identified with corresponding changes in height. □

The Strong Kleene component provides the clearest finite example in the release of why size and height must be computed independently.

5.9 The BCK-Related Union Component

The BCK-related analysis produced a substantially larger binary function union:

$$| C_{\mathrm{BCK}} | = 1152.$$

Across the tested preserving singleton presentations, the saturation-size range was

$$10 \leq S_{\max} \leq 13.$$

The saturation height was constant:

$$H_{\max} = 5.$$

Thus

$$\omega_s = 3$$

while

$$\omega_h = 0.$$

Every tested primitive presentation reached the entire component within height five, yet some required substantially more total operation nodes in their most difficult minimum-size representations.

This is a direct separation between size sensitivity and height rigidity.

Proposition 5.4. A component may be presentation-sensitive in exact minimum size while presentation-invariant in saturation height.

Proof. The BCK-related union component has varying saturation size and constant saturation height across the tested preserving presentations. □

Height invariance does not imply full profile invariance. The identities of functions appearing at each height may still differ even when the maximum height is fixed.

Likewise, constant saturation height does not imply constant total height:

$$H_{\max}(g) = H_{\max}(h)$$

does not force

$$U(g) = U(h).$$

The maximum is only one statistic.

5.10 Profile Stratification

For a generator g, define the exact-size strata

$$C_k^s(g) = \{f \in C : s_g(f) = k\}$$

and the cumulative filtration

$$C_{\leq k}^s(g) = \{f \in C : s_g(f) \leq k\}.$$

Similarly, define

$$C_k^h(g) = \{f \in C : h_g(f) = k\}$$

and

$$C_{\leq k}^h(g) = \{f \in C : h_g(f) \leq k\}.$$

Then

$$C_{\leq 0}^s(g) \subseteq C_{\leq 1}^s(g) \subseteq \cdots \subseteq C$$

and

$$C_{\leq 0}^h(g) \subseteq C_{\leq 1}^h(g) \subseteq \cdots \subseteq C.$$

The final indices are

$$S_{\max}(g)$$

and

$$H_{\max}(g),$$

respectively.

Primitive substitution changes these filtrations.

Even when two generators have equal total costs, they may make different functions accessible early:

$$C_{\leq k}^s(g) \neq C_{\leq k}^s(h).$$

The profiles therefore encode an ordering of conceptual accessibility.

A function is early relative to $g$when

$$s_g(f)$$

or

$$h_g(f)$$

is small.

It is late relative to $g$when the corresponding value is near saturation.

No function is intrinsically early or late independently of presentation.

5.11 Relative Accessibility

For generators $g, h \in G(C, u)$, define the relative size displacement of $f$by

$$D^s_{g,h}(f) = s_g(f) - s_h(f).$$

Define the relative height displacement by

$$D^h_{g,h}(f) = h_g(f) - h_h(f).$$

Then

$$D^s_{g,h}(f) > 0$$

means that $f$is cheaper from $h$than from g.

Likewise,

$$D^s_{g,h}(f) < 0$$

means that $f$is cheaper from g.

The displacement is antisymmetric:

$$D^s_{g,h}(f) = -D^s_{h,g}(f).$$

Summing over the component gives

$$T(g) - T(h) = \sum_{f \in C} D^s_{g,h}\,(f).$$

Thus a global advantage may arise from:

many small savings,
a few large savings,

or

a mixture of savings and losses with positive net balance.

Two presentations may be incomparable pointwise even when one has a lower total cost.

Definition 5.2. The size-win set of $g$over $h$is

$$W^s(g,h) = \{f \in C : s_g(f) < s_h(f)\}.$$

The tie set is

$$E^s(g,h) = \{f \in C : s_g(f) = s_h(f)\}.$$

The loss set is

$$L^s(g,h) = \{f \in C : s_g(f) > s_h(f)\}.$$

These three sets partition C.

An analogous partition is defined for height.

A full comparison between presentations therefore includes not only

$$T(g)\text{and}T(h),$$

but also the distribution of local gains and losses.

5.12 Primitive Centrality

A generator that makes many functions inexpensive occupies a central position in the formation geometry.

Define the mean size radius of $g$by

$$\bar{s}(g) = \frac{T(g)}{|\,C\,|}.$$

Define the mean height radius by

$$\bar{h}(g) = \frac{U(g)}{|\ C\ |}.$$

A size-central generator minimizes

$$\bar{s}(g),$$

equivalently $T(g)$, over $G(C, u)$.

A height-central generator minimizes

$$\bar{h}(g).$$

A minimax-size generator minimizes

$$S_{\max}(g).$$

A minimax-height generator minimizes

$$H_{\max}(g).$$

These optimizers need not coincide.

Definition 5.3. The singleton center set is

$$\mathrm{Cent}_s(C, u) = \arg\min_{g \in G(C,u)} T(g).$$

The singleton minimax set is

$$\mathrm{Rad}_s(C, u) = \arg\min_{g \in G(C,u)} S_{\max}(g).$$

Corresponding height sets are defined by $U$and $H_{\max}$.

The terminology is geometric: the primitive is treated as an access origin, and the component is measured relative to it.

5.13 Presentation Entropy

The size histogram may be normalized to a probability distribution:

$$p_k^S(g) = \frac{N_k^S(g)}{\mid C \mid}.$$

Define the size-profile entropy

$$\mathcal{H}_s(g) = -\sum_k p_k^s\,(g)\log p_k^s(g).$$

Similarly,

$$\mathcal{H}_h(g) = -\sum_k p_k^h\,(g)\log p_k^h(g).$$

These quantities measure dispersion across complexity layers, not mathematical uncertainty.

A low-entropy profile concentrates many functions at a small number of costs. A higher-entropy profile spreads the component across more layers.

The release does not use entropy as its primary optimization criterion. It is included as a natural derived invariant of the exact profiles.

The essential point is that the finite profile supports multiple analyses once exact minimum costs have been computed.

5.14 Profile Equivalence

Two generators $g, h \in G(C, u)$may generate the same component and even have the same aggregate statistics without having identical pointwise profiles.

Define:

$$g \equiv_{\text{agg}} h$$

when

$$T(g) = T(h),$$
$$U(g) = U(h),$$
$$S_{\max}(g) = S_{\max}(h),$$

and

$$H_{\max}(g) = H_{\max}(h).$$

Define:

$$g \equiv_{\text{hist}} h$$

when their size and height histograms agree.

Define:

$$g \equiv_{\text{prof}} h$$

when

$$s_g(f) = s_h(f)$$

and

$$h_g(f) = h_h(f)$$

for every $f \in C$.

Then

$$g \equiv_{\text{prof}} h \Longrightarrow g \equiv_{\text{hist}} h \Longrightarrow g \equiv_{\text{agg}} h.$$

The reverse implications need not hold.

This hierarchy distinguishes increasingly coarse notions of presentation equivalence.

5.15 Conjugacy and Profile Invariance

Carrier relabeling differs fundamentally from polarity transformation.

Let $\rho: A \to A$be a carrier permutation. If $u$and $g$are simultaneously relabeled,

$$u^{\rho} = \rho u \rho^{-1},$$
$$g^{\rho}(x, y) = \rho(g(\rho^{-1}(x), \rho^{-1}(y))),$$

then every term is transferred without changing its syntax tree.

Therefore

$$s_{g^\rho}(f^\rho) = s_g(f)$$

and

$$h_{g^\rho}(f^\rho) = h_g(f).$$

Carrier conjugacy is profile-preserving up to permutation of function labels.

Polarity transformation, by contrast, changes where unary nodes occur inside term representations. It preserves content but may change cost.

Thus:

carrier relabeling ⇒ profile isomorphism,

whereas

polarity interdefinition ⇒ component equality only.

This distinction prevents symmetries of the operation space from being treated as automatic cost symmetries.

5.16 Formation Profiles as Algebraic Data

Classical finite-algebra records commonly include:

carrier cardinality,
operation tables,
identities,
subalgebras,
congruences,

and

automorphisms.

The present framework adds presentation-relative data:

minimum term-size profile,
minimum height profile,
generator spectrum,
promotion gain,

and

refactoring graph.

These do not replace classical invariants. They describe the cost structure induced by a selected language.

The data become especially relevant when primitive choice is itself produced by an automated discovery process. A system that discovers a component must still decide how to present it.

A discovered theory may have many extensionally equivalent descriptions. The formation spectrum records their exact differences.

5.17 Cycle 7 Result

Cycle 7 establishes the following.

Theorem 5.2. Binary-component equality does not determine a unique exact formation profile.

More precisely, in each of the Boolean, Weak Kleene, Strong Kleene, and BCK-related reference components, there exist preserving singleton signatures with unequal saturation-size profiles. In the first three components, preserving singleton signatures also attain unequal saturation heights.

The theorem converts the post-birth language problem into a nontrivial finite optimization problem.

If every generator produced the same profile, signature refactoring would have no complexity content. The observed profile variation shows that primitive selection changes the exact architecture of access.

The result may be summarized as

same functions ⇒/same formation.

The next stage asks whether the profile differences separate genuine theory births from structurally defective or duplicated candidates.

6. Genuine and False-Birth Complexity Spectra

6.1 Complexity After Structural Certification

A finite system may exhibit a large identity set, reproduce a known bounded equational profile, and generate a substantial term-function component while still failing the structural conditions required for the intended algebraic interpretation.

Exact formation complexity therefore cannot serve as a substitute for structural certification.

Let Abe a finite algebra in a signature containing a fixed unary operation $u$and one or more binary operations. Let

$$C_2(A)$$

denote its binary term-function component, and let

$$\Psi(A;S)$$

denote the exact formation profile induced by a preserving primitive signature S.

A rich profile may establish that the algebra supports nontrivial internal generation. It does not establish that the algebra belongs to a proposed variety, satisfies an intended quasi-identity, supports a compatible order, or admits a well-defined quotient repair.

The certification sequence must therefore remain ordered:

$$\text{equational recurrence} \longrightarrow \text{structural admissibility} \longrightarrow \text{component formation} \longrightarrow \text{complexity spectrum.}$$

A complexity spectrum computed before structural admissibility remains mathematically valid as a property of the finite operation system. It is not yet evidence for the intended named theory.

Definition 6.1. A genuine finite birth relative to a declared certificate $\mathcal{C}$is a finite algebra Asatisfying every condition in $\mathcal{C}$.

A false birth is a candidate Athat passes the bounded equational stage but fails at least one required structural condition.

The adjective false is certificate-relative. It does not mean that the operation table is invalid or that its term-function component is nonexistent. It means that the proposed interpretation has not survived certification.

6.2 Equational Agreement and Structural Failure

Let $E_d$be a bounded observational identity set. Suppose two finite algebras $A$and Bsatisfy

$$\mathrm{Eq}_d(A) = \mathrm{Eq}_d(B).$$

This equality does not imply that $A$and Bagree on non-equational properties.

For example, a relation may be induced by a binary operation:

$$x \preceq y \Leftrightarrow x * y = 0.$$

The intended structure may require antisymmetry:

$$x \preceq y \wedge y \preceq x \Longrightarrow x = y.$$

Antisymmetry is not itself an equation in the original operational signature.

Thus two algebras may agree on all tested equations while differing on whether the induced relation is a partial order.

Proposition 6.1. Equality of bounded equational profiles does not imply equality of structural birth certificates.

Proof. A birth certificate may contain quasi-identities, relational properties, cardinality conditions, quotient-compatibility requirements, or external interpretations not expressible as equations in the tested signature. Therefore bounded equational equality leaves those conditions undetermined. □

The Stage II BCK-related duplicate exhibited precisely this separation.

It reproduced the relevant bounded equational behavior but failed antisymmetry of the induced relation.

6.3 Quotient Repair

Suppose a finite candidate fails antisymmetry because distinct elements are mutually related. A natural repair is to define

$$x \sim y \Leftrightarrow x \preceq y \wedge y \preceq x.$$

The relation $\sim$may be an equivalence relation. This alone does not ensure that the quotient carries the original algebraic operations.

For a binary operation $*$, compatibility requires

$$x \sim x', y \sim y'$$

to imply

$$x * y \sim x' * y'.$$

Equivalently, $\sim$must be a congruence.

Definition 6.2. A quotient repair is admissible when the proposed identification relation is a congruence for every operation retained in the quotient signature.

If compatibility fails, the expression

$$[x] * [y] = [x * y]$$

depends on the chosen representatives and is therefore not well defined.

Proposition 6.2. Failure of quotient compatibility prevents a structurally defective finite candidate from being repaired by identifying mutually related elements.

Proof. If there exist

$$x \sim x', y \sim y'$$

with

$$x * y \not\sim x' * y',$$

then the classes $[x] = [x']$and $[y] = [y']$produce two different output classes. Hence the quotient operation is not well defined. □

The BCK-related duplicate failed both the original antisymmetry condition and the natural quotient-compatibility repair.

Its negative certificate was therefore decisive.

6.4 Complexity of a False Birth

Let A_F be a false birth and let

$$C_F = C_2(A_F)$$

be its binary term-function component.

The component C_F may still possess:

singleton generators,
exact size profiles,
exact height profiles,
promotion gains,

and

preserving signature landscapes.

These quantities describe the finite operation system A_F. They do not certify the rejected interpretation.

Thus the statement

$$S_{\max}(A_F) = k$$

may be correct even when the statement

A_F is a valid finite model of the proposed theory

is false.

The two judgments concern different objects.

The first concerns internal term generation.

The second concerns theory membership or structural admissibility.

Definition 6.3. The intrinsic complexity spectrum of a finite operation system is the collection of exact presentation profiles computed solely from its operation tables.

The certified theory spectrum is the intrinsic spectrum interpreted only after the system passes the declared birth certificate.

This distinction prevents complexity from laundering a failed theory claim.

6.5 Genuine and False-Birth Spectra

Let A_G be a certified genuine finite birth and A_F a structurally rejected duplicate.

Suppose their binary components satisfy

$$| C_2(A_G) | = | C_2(A_F) |$$

or have overlapping bounded identity profiles.

Their formation spectra may nevertheless differ.

Define

$$\mathfrak{F}_1(A) = \{\Psi(g) : g \in G(C_2(A), u)\}.$$

A spectrum comparison may examine:

$$| G(C_2(A), u) |,$$
$$\min_g T(g),$$
$$\max_g T(g),$$
$$\min_g S_{\max}(g),$$
$$\max_g S_{\max}(g),$$
$$\min_g H_{\max}(g),$$

and

$$\max_g H_{\max}(g).$$

The spectra may separate candidates that bounded equation counts do not.

This separation is diagnostic rather than dispositive. Different spectra prove that the finite generative organizations differ. Equal spectra would not prove structural equivalence.

Proposition 6.3. Formation-spectrum inequality implies presentation-sensitive generative inequality but does not, by itself, determine theory membership.

Proof. A formation spectrum is computed from term representations in the finite component. Structural theory membership may depend on additional equations, quasi-identities, relations, congruences, or higher-arity operations. Therefore spectrum inequality is informative but not sufficient for classification. □

6.6 The BCK-Related Union Spectrum

The BCK-related computations produced a binary union component of cardinality

$$| C_{\mathrm{BCK}} |= 1152.$$

Across the tested preserving singleton presentations,

$$10 \leq S_{\max} \leq 13,$$

while

$$H_{\max} = 5.$$

Hence the size-radius spread is

$$13 - 10 = 3,$$

and the height-radius spread is

$$5 - 5 = 0.$$

The result shows that alternative primitive presentations substantially alter minimum operation-node requirements while leaving maximum minimum depth unchanged.

The spectrum therefore contains nontrivial presentation geometry even though one candidate interpretation was rejected.

This is scientifically useful for two reasons.

First, it confirms that the computational machinery is not restricted to very small components such as the Boolean and Kleene examples.

Second, it demonstrates that structural rejection does not force computational triviality.

A false birth may possess a rich intrinsic algebraic organization.

What fails is the proposed identity of that organization with the intended theory.

6.7 Certificate Layers

A complete candidate record should separate at least four layers.

The first layer is extensional:

$$\mathcal{E}(A) = \text{operation tables and term functions.}$$

The second is equational:

$$\mathcal{Q}_d(A) = \mathrm{Eq}_d(A).$$

The third is structural:

$$\mathcal{S}(A) = \text{non-equational certificate conditions.}$$

The fourth is generative:

$$\mathcal{G}(A) = \text{exact formation and refactoring spectra.}$$

These layers answer different questions.

$\mathcal{E}(A)$:What finite functions exist?
$\mathcal{Q}_d(A)$:Which bounded identities hold?
$\mathcal{S}(A)$:Does the intended structure exist?
$\mathcal{G}(A)$:How is the component generated under alternative signatures?

A birth certificate should not collapse these layers into one score.

Definition 6.4. A layered birth record is a tuple

$$\mathfrak{B}(A) = (\mathcal{E}(A), \mathcal{Q}_d(A), \mathcal{S}(A), \mathcal{G}(A)).$$

The candidate is certified only if the structural layer returns an admissible decision.

The generative layer remains reportable under either decision, provided its interpretation is stated correctly.

6.8 Negative Certificates as Mathematical Output

A rejected candidate can contribute more than a discarded search path.

A negative certificate may contain:

the shortest failing witness,
the exact elements violating antisymmetry,
the proposed quotient relation,
the exact compatibility counterexample,

and

the intrinsic component spectrum.

Such a record identifies why an equation-rich finite system fails to instantiate the intended structure.

Definition 6.5. A minimal structural obstruction is a smallest finite witness, under a declared ordering, to failure of a certificate condition.

For antisymmetry, an obstruction is a pair

$$(x, y)$$

with

$$x \neq y,$$
$$x \preceq y,$$

and

$$y \preceq x.$$

For quotient compatibility, an obstruction is a quadruple

$$(x, x', y, y')$$

such that

$$x \sim x', y \sim y',$$

but

$$x * y \not\sim x' * y'.$$

The negative certificate is exact and independently verifiable.

It therefore belongs to the mathematical output of the search.

6.9 Explanation-Subtracted Complexity

The same principle used for identity discovery applies to formation complexity.

Suppose a large reduction in total term size follows almost entirely from promoting one expensive derived operation. Reporting only the final cost improvement may exaggerate the novelty of the signature change.

Let $m \in C$ be promoted from a preserving singleton signature S.

The direct gain is

$$D_T(S, m) = s_S(m) - 1.$$

The propagated gain is

$$P_T(S, m) = \sum_{f \neq m} \big(s_S(f) - s_{S \cup \{m\}}(f)\big).$$

The total gain is

$$\Delta_T(S, m) = D_T(S, m) + P_T(S, m).$$

The direct term is explained by declaring m primitive. The propagated term measures compression of the remainder of the component.

This decomposition is the complexity analogue of explanation subtraction.

Definition 6.6. The residual promotion gain is

$$R_T(S, m) = P_T(S, m).$$

A promotion with

$$R_T(S, m) = 0$$

compresses only the promoted function itself.

A promotion with large

$$R_T(S, m)$$

reorganizes access to many other functions.

This distinction becomes important when evaluating whether a refactoring reveals a globally useful operation or merely assigns zero derivational burden to one isolated term.

6.10 Local and Global Macros

A promoted primitive $m$is local when most of its gain is direct:

$$D_T(S, m) \gg P_T(S, m).$$

It is global when the propagated gain dominates:

$$P_T(S, m) \gg D_T(S, m).$$

Define the propagation ratio

$$\rho_T(S, m) = \frac{P_T(S, m)}{\Delta_T(S, m)}$$

when

$$\Delta_T(S, m) > 0.$$

Then

$$0 \leq \rho_T(S, m) \leq 1.$$

A value near zero indicates mostly local compression.

A value near one indicates that the promoted primitive is repeatedly reused across the component.

This ratio refines the interpretation of primitive promotion.

The Stage 2 release primarily reports exact total gains and optimal signatures. The decomposition supplies a natural extension for future analysis.

6.11 Spectrum Distance

To compare two finite systems $A$and $B$with components of equal cardinality, one may sort their generator statistics and define a spectrum distance.

Let

$$\mathbf{v}(A)$$

be a canonically ordered list of aggregate formation descriptors, for example

$$\mathbf{v}(A) = \mathrm{sort}\{(T(g), S_{\max}(g), U(g), H_{\max}(g)) : g \in G(C_2(A), u)\}.$$

When the lists have equal length, define

$$d_{\mathrm{spec}}(A, B) = \| \mathbf{v}(A) - \mathbf{v}(B) \|_1.$$

More generally, histogram or transport distances may compare unequal generator sets.

This construction is not required for the current finite certification results. It formalizes the idea that two equation-similar systems may occupy different regions of generative-profile space.

Proposition 6.4. Spectrum distance is invariant under simultaneous carrier relabeling when the canonical ordering is relabeling-invariant.

Proof. Carrier relabeling preserves each exact size and height profile up to permutation of function labels. Aggregate descriptors are therefore unchanged. □

Spectrum distance may consequently serve as an isomorphism-respecting diagnostic.

6.12 False-Birth Neighborhoods

A false birth need not be isolated. It may lie near genuine models under several finite comparisons:

identity overlap,
operation-table distance,
component cardinality,

or

formation-spectrum distance.

Let $\mathcal{P}$ be a finite population of candidate systems. Define a graph whose vertices are systems and whose edges join candidates with high residual identity overlap.

A false birth may occupy a dense cluster adjacent to genuine models. Its rejection may depend on one short obstruction invisible to the clustering score.

This creates a useful adversarial setting:

maximal equational similarity + minimal structural defect.

Such candidates are stronger tests of certification than obviously degenerate operations.

The BCK-related duplicate belongs conceptually to this class. It was not rejected because it lacked internal algebraic organization. It was rejected because a decisive structural requirement failed.

6.13 Birth Validity and Presentation Quality

A candidate may be classified along two independent axes.

The validity axis is

$$V(A) \in \{0,1\},$$

where

$$V(A) = 1$$

means that the structural birth certificate passes.

The presentation-quality axis may be represented by

$$Q(A,S) = (T(S), S_{\max}(S), U(S), H_{\max}(S)).$$

This produces four conceptual regions.

$V(A)$	$Q(A,S)$	Interpretation
1	strong	certified theory with efficient presentation
1	weak	certified theory with inefficient presentation
0	strong	rich false birth
0	weak	structurally invalid and generatively poor candidate

The third region is methodologically important.

A rich false birth shows why discovery systems require more than complexity maximization.

6.14 No Complexity Threshold for Validity

There is no universal scalar K such that

$$T(S) < K$$

or

$$S_{\max}(S) < K$$

implies structural validity.

Likewise, high complexity does not imply invalidity.

Theorem 6.1. Structural birth validity cannot be reduced to a threshold on exact minimum term-size or height statistics.

Proof. Structural validity depends on satisfaction of the declared certificate. Complexity statistics measure representation cost within a generated component. A structurally invalid candidate may have a compact and highly organized component, while a valid algebra may require a comparatively expensive primitive presentation. Therefore no general implication exists in either direction. □

Complexity is a secondary invariant after certification, not a replacement for certification.

6.15 Genuine Formation Spectra

For a certified family $\mathcal{V}$, let

$$\mathcal{V}_{\text{fin}}$$

denote its finite members within a declared carrier bound.

A genuine formation spectrum may be defined at three levels.

The model-level spectrum is

$$\mathfrak{F}(A)$$

for one certified finite algebra A.

The isomorphism-class spectrum is

$$\mathfrak{F}([A]),$$

which is well defined because exact profiles are preserved by carrier relabeling.

The family-level spectrum is

$$\mathfrak{F}(\mathcal{V}; n) = \bigcup_{\substack{A \in \mathcal{V}_{\mathrm{fin}} \\ |A| = n}} \mathfrak{F}(A).$$

Such spectra may reveal whether one finite model is atypical or representative within a variety.

The current release focuses on selected canonical finite components rather than a complete family-level census.

The definitions indicate how the analysis may be extended.

6.16 False-Birth Spectra as Countertheory Data

A false birth can be treated as a finite countertheory.

It may satisfy a substantial fragment of a target theory while failing one decisive condition. Its intrinsic spectrum then describes how much algebraic organization can exist without the missing condition.

Suppose $E$is the recovered equational fragment and $P$the failed structural property.

The false birth demonstrates consistency, at finite observational scale, of

$$E + \neg P.$$

Its formation spectrum describes the generative complexity of that countertheory.

This is stronger than a bare counterexample. It records not only that Pfails, but how an entire finite operational ecology forms around the failure.

Definition 6.7. A spectrum-certified countermodel to Punder $E$is a finite algebra Asuch that:

$$A \vDash E,$$
$$A \nvDash P,$$

and

$$\mathfrak{F}(A)$$

has been computed exactly.

The BCK-related duplicate provides a prototype.

6.17 Cycle 8 Result

Cycle 8 compared exact formation spectra across genuine and false-birth systems and retained structural rejection as part of the final evidence.

The principal conclusion is not that complexity alone distinguishes every valid theory from every false birth.

The conclusion is that the discovery architecture can preserve both judgments simultaneously:

this candidate has a rich exact formation spectrum,

and

this candidate fails the proposed structural birth certificate.

These statements are compatible.

Theorem 6.2. Exact generative richness does not entail structural admissibility.

The BCK-related duplicate supports a binary union component of cardinality

$$1152$$

with preserving singleton saturation sizes between

$$10$$

and

$$13,$$

yet the proposed BCK interpretation fails antisymmetry and quotient compatibility.

Therefore

rich component ⇒/valid birth.

The result strengthens the rejection discipline of Algebraic Abiogenesis.

The system does not equate:

many identities,
large term-function components,
compact formation profiles,

or

recognizable resemblance

with successful theory birth.

A successful birth requires all declared certificate layers to agree.

The next stage turns from classification to intervention. Given a certified component and an exact formation profile, derived operations are promoted recursively to determine how the theory's access geometry changes under conservative enrichment.

7. Recursive Primitive Promotion

7.1 Conservative Enrichment

Let (C, u)be a finite pointed binary term-function component, and let

$$S \in \mathcal{R}(C, u)$$

be a preserving signature. Every function

$$m \in C$$

is already term-definable from $u$and S. Declaring $m$primitive therefore changes presentation without changing binary extensional content.

Define the promotion operator

$$\mathrm{Prom}_m(S) = S \cup \{m\}.$$

Since

$$m \in C_2(S; u),$$

monotonicity gives

$$C_2(S; u) \subseteq C_2(S \cup \{m\}; u).$$

Because every added primitive lies in C,

$$C_2(S \cup \{m\}; u) \subseteq C.$$

Hence

$$C_2(S \cup \{m\}; u) = C.$$

Proposition 7.1. Primitive promotion is conservative with respect to the designated binary component:

$$S \in \mathcal{R}(C, u), m \in C \Longrightarrow S \cup \{m\} \in \mathcal{R}(C, u).$$

Promotion creates no new binary term function. It changes only the primitive cost structure.

This distinction separates promotion from content-generating extension. If $m \notin C$, adjoining $m$may enlarge the component. Such an extension is not a refactoring of C.

7.2 Pointwise Monotonicity

Let

$$S^+ = S \cup \{m\}.$$

Every S-term remains an S^+-term. Therefore no function becomes more expensive.

Theorem 7.1. For every $f \in C$,

$$s_{S^+}(f) \leq s_S(f)$$

and

$$h_{S^+}(f) \leq h_S(f).$$

Consequently,

$$T(S^+) \leq T(S),$$
$$U(S^+) \leq U(S),$$
$$S_{\max}(S^+) \leq S_{\max}(S),$$

and

$$H_{\max}(S^+) \leq H_{\max}(S).$$

Proof. The admissible representation set for f under S is contained in the admissible representation set under S^+. Minimum values over a larger set cannot increase. Summation and maximization preserve the inequalities. □

Promotion is therefore monotone in both exact complexity geometries.

It need not strictly improve either profile. A promoted operation may already have primitive cost one or may fail to shorten any minimum representation.

Definition 7.1. A promotion m is size-effective relative to S when

$$T(S \cup \{m\}) < T(S).$$

It is height-effective when

$$U(S \cup \{m\}) < U(S).$$

It is saturation-effective when it lowers at least one of

$$S_{\max} \text{ or } H_{\max}.$$

A promotion may be effective under one criterion and ineffective under another.

7.3 Direct and Propagated Gain

The promoted operation acquires primitive size at most one:

$$s_{S^+}(m) = 1$$

unless $m$is a projection, in which case its size remains zero.

Assume $m$is nonprojective. Its direct size reduction is

$$D_T(S, m) = s_S(m) - 1.$$

For every $f \in C$, define the pointwise promotion gain

$$\delta^S_{S,m}(f) = s_S(f) - s_{S^+}(f).$$

Then

$$\delta^S_{S,m}(f) \geq 0.$$

The total gain is

$$\Delta_T(S, m) = T(S) - T(S^+) = \sum_{f \in C} \delta^S_{S,m}(f).$$

Separate the gain on the promoted operation from gains propagated to the rest of the component:

$$\Delta_T(S, m) = D_T(S, m) + P_T(S, m),$$

where

$$P_T(S, m) = \sum_{f \in C \setminus \{m\}} \delta^S_{S,m}(f).$$

The propagated gain measures how often $m$functions as a reusable macro inside minimum representations of other operations.

Similarly, define

$$D_U(S, m) = h_S(m) - 1$$

and

$$P_U(S, m) = \sum_{f \neq m} (h_S(f) - h_{S^+}(f)).$$

Then

$$\Delta_U(S, m) = D_U(S, m) + P_U(S, m).$$

Definition 7.2. The size-propagation ratio is

$$\rho_T(S, m) = \frac{P_T(S, m)}{\Delta_T(S, m)}$$

whenever

$$\Delta_T(S, m) > 0.$$

A large value of ρ_T indicates that the promoted operation acts as a global macro rather than merely compressing its own representation.

7.4 Promotion Ranking

A one-step promotion policy assigns a score to each candidate

$$m \in C \setminus S.$$

Natural scores include:

$$\Delta_T(S, m),$$
$$\Delta_U(S, m),$$
$$\Delta_{\max}(S, m) = S_{\max}(S) - S_{\max}(S \cup \{m\}),$$
$$\Delta_H(S, m) = H_{\max}(S) - H_{\max}(S \cup \{m\}),$$

and the efficiency ratio

$$\eta_\lambda(S,m) = \Delta_T(S,m) - \lambda,$$

where $\lambda$is the acquisition price of one additional primitive.

A size-greedy promotion selects

$$m^* \in \arg\max_{m \in C \setminus S} \Delta_T(S,m).$$

A height-greedy promotion selects

$$m_h^* \in \arg\max_{m \in C \setminus S} \Delta_U(S,m).$$

A minimax promotion selects an operation maximizing the decrease in saturation size or height.

The resulting choices need not agree.

Proposition 7.2. A maximizer of $\Delta_T$need not maximize $\Delta_{\max}$, and a maximizer of $\Delta_U$need not maximize Δ_H.

Proof. Total gain aggregates improvements over all functions, whereas saturation gain depends only on the most expensive function or functions. A promotion may produce small savings across many functions without affecting the maximum, while another may shorten one extremal function and leave the total nearly unchanged. □

Primitive promotion is therefore intrinsically multiobjective.

7.5 Recursive Promotion Sequence

Starting from a preserving signature S_0, define recursively

$$S_{k+1} = S_k \cup \{m_k\},$$

where

$$m_k \in \arg\max_{m \in C \setminus S_k} \Delta_T(S_k,m)$$

under a frozen tie-breaking rule.

This yields the chain

$$S_0 \subseteq S_1 \subseteq S_2 \subseteq \cdots \subseteq C.$$

Every stage preserves the component:

$$C_2(S_k; u) = C.$$

The total-size sequence is nonincreasing:

$$T(S_0) \geq T(S_1) \geq T(S_2) \geq \cdots .$$

The same holds for total height and both saturation radii.

Because C is finite, the sequence terminates after at most

$$| C | - | S_0 |$$

strict additions.

Definition 7.3. A recursive promotion trajectory is a finite chain

$$\mathcal{P}(S_0) = (S_0, S_1, \ldots, S_r)$$

generated by a declared promotion policy.

The trajectory records how a presentation becomes progressively enriched while preserving content.

7.6 Termination

Theorem 7.2. Every recursive promotion process on a finite component terminates.

Proof. At each strict step, one previously absent operation is added to the finite set $S_k \subseteq C$. No operation is removed. Therefore after at most $| C | - | S_0 |$additions, no further candidate remains. □

Termination does not imply that every intermediate step is useful.

A policy may continue adding operations even after

$$\Delta_T(S_k, m) = 0$$

for every remaining m.

Define the size-stopping index

$$r_T = \min\left\{k: \max_{m \in C \setminus S_k} \Delta_T(S_k, m) = 0\right\}.$$

At S_{r_T}, no single additional primitive lowers total minimum size.

Definition 7.4. A preserving signature $S$is one-step size-stable when

$$\Delta_T(S, m) = 0$$

for every $m \in C \setminus S$.

It is one-step height-stable when

$$\Delta_U(S, m) = 0$$

for every remaining m.

One-step stability is local relative to single primitive additions. It does not imply global optimality under arbitrary signature replacement.

7.7 Greedy Optimality and Its Limits

The greedy rule chooses the best immediate promotion. It does not generally guarantee the best width-k signature.

Let

$$S_0$$

be fixed. For a promotion budget r, define the globally optimal extension

$$S_r^* \in \arg\min\{T(S): S_0 \subseteq S \subseteq C, \ \mid S \mid = \mid S_0 \mid + r\}.$$

The greedy trajectory produces S_r^{gr}.

In general,

$$T(S_r^{\mathrm{gr}}) \geq T(S_r^*).$$

Equality is not automatic.

The obstruction is interaction among promoted operations. One operation may have modest immediate gain but combine with another to produce a larger joint reduction than the initially best macro.

Definition 7.5. The joint promotion gain of $m, n \notin S$is

$$\Delta_T(S; \{m, n\}) = T(S) - T(S \cup \{m, n\}).$$

The interaction term is

$$I_T(S; m, n) = \Delta_T(S; \{m, n\}) - \Delta_T(S, m) - \Delta_T(S, n).$$

If

$$I_T > 0,$$

the pair has superadditive compression.

If

$$I_T < 0,$$

their gains overlap.

If

$$I_T = 0,$$

their total effects are additive.

Because promotion gains may interact, a greedy path need not be globally optimal.

The Stage 2 release therefore treats recursive promotion as an exact heuristic trajectory, not as a proof of globally optimal multi-primitive design.

7.8 Diminishing Returns

A natural question is whether promotion gain satisfies diminishing returns:

$$S \subseteq T \Longrightarrow \Delta_T(S, m) \geq \Delta_T(T, m).$$

If this held universally, the objective

$$-T(S)$$

would behave submodularly, and greedy optimization would inherit strong approximation guarantees.

No such general theorem is assumed.

An operation may become more useful after another macro has been promoted because the two can compose efficiently. Thus

$$\Delta_T(S, m) < \Delta_T(T, m)$$

is possible in principle.

Proposition 7.3. Monotonicity of Tunder primitive addition does not imply diminishing returns of promotion gain.

Proof. Monotonicity states only that

$$T(S \cup \{m\}) \leq T(S).$$

It imposes no ordering between the marginal reductions measured from two different base signatures $S \subseteq T$. Interaction among macros may increase or decrease marginal utility. □

Submodularity is therefore an empirical or theorem-level property to be tested for particular components, not a default assumption.

7.9 Promotion Graph

Let the vertices be preserving signatures in a bounded width family:

$$\mathcal{R}_{\leq k}(C, u).$$

Draw a directed edge

$$S \longrightarrow S \cup \{m\}$$

whenever

$$m \in C \setminus S$$

and

$$| S |< k.$$

Assign the edge weights

$$w_T(S, m) = \Delta_T(S, m),$$
$$w_U(S, m) = \Delta_U(S, m),$$

or acquisition-adjusted gain

$$w_\lambda(S, m) = \Delta_T(S, m) - \lambda.$$

The resulting directed acyclic graph is the promotion graph

$$\mathbf{P}_k(C, u).$$

A recursive promotion trajectory is a directed path in $\mathbf{P}_k(C, u)$.

Because edges only increase signature width, cycles are impossible.

The promotion graph is a subgraph of the broader refactoring landscape, which may also contain primitive removals and replacements.

7.10 Promotion Under Primitive Cost

Without an acquisition penalty, every effective primitive addition is favorable.

To charge for vocabulary expansion, define

$$J_\lambda(S) = T(S) + \lambda(| S | -1).$$

Promoting $m$changes the objective by

$$J_\lambda(S \cup \{m\}) - J_\lambda(S) = -\Delta_T(S, m) + \lambda.$$

Therefore the promotion is favorable exactly when

$$\Delta_T(S, m) > \lambda.$$

It is neutral when

$$\Delta_T(S, m) = \lambda$$

and unfavorable when

$$\Delta_T(S, m) < \lambda.$$

Proposition 7.4. Under primitive price λ, the rational one-step promotion rule is

$$m^* \in \arg\max_{m \in C \setminus S} (\Delta_T(S, m) - \lambda),$$

with promotion performed only when the maximum is positive.

This gives a principled stopping condition:

$$\max_m \Delta_T(S, m) \leq \lambda.$$

The vocabulary stops growing when no remaining macro saves more than it costs.

7.11 Break-Even Price of a Macro

For a candidate m, define its break-even primitive price by

$$\lambda^*(S, m) = \Delta_T(S, m).$$

The promotion is justified for

$$\lambda < \lambda^*(S, m).$$

The break-even value includes both direct and propagated savings:

$$\lambda^*(S, m) = D_T(S, m) + P_T(S, m).$$

A macro with small intrinsic cost but high propagated reuse can therefore have a large break-even price.

This formalizes the distinction between a rarely used abbreviation and a structurally central primitive.

7.12 Height-Aware Promotion

A size-only promotion policy may increase vocabulary to reduce total operation count while leaving the critical depth unchanged.

A combined objective may be defined by

$$J_{\lambda,\mu,\nu}(S) = T(S) + \lambda(|\,S\,| - 1) + \mu S_{\max}(S) + \nu H_{\max}(S),$$

where

$$\lambda, \mu, \nu \geq 0.$$

Alternatively, total height may be included:

$$J_{\lambda,\alpha,\beta,\gamma}(S) = T(S) + \alpha U(S) + \beta S_{\max}(S) + \gamma H_{\max}(S) + \lambda(|\,S\,| - 1).$$

The optimal promotion then depends on the declared use model.

A symbolic theorem prover may value total expression size.

A parallel circuit model may value height.

A bounded-depth architecture may prioritize $H_{\max}$.

A compact mathematical language may impose a high λ.

No single cost function is universally canonical.

7.13 Promotion Witnesses

For each effective promotion m, the computational record should identify:

$$s_S(m),$$
$$h_S(m),$$
$$\Delta_T(S, m),$$
$$\Delta_U(S, m),$$
$$\Delta_{\max}(S, m),$$
$$\Delta_H(S, m),$$

and the set

$$A_m = \{f \in C : \delta^S_{S,m}(f) > 0\}.$$

The set $A_m$is the promotion influence region.

Define its cardinality by

$$a_m = | A_m |.$$

A macro may have:

large gain and small influence region,

or

small per-function gain and large influence region.

The influence record distinguishes these cases.

Definition 7.6. The mean gain over affected functions is

$$\bar{\delta}_m = \frac{\Delta_T(S, m)}{| A_m |}$$

when $A_m \neq \emptyset$.

The pair

$$\left(| A_m |, \bar{\delta}_m\right)$$

describes the breadth and intensity of promotion.

7.14 Recursive Promotion as Language Formation

At the start of the promotion process, the theory is expressed through a narrow primitive vocabulary.

Each step identifies a derived operation whose explicit promotion most compresses the current component.

The resulting sequence is

$$S_0 \subset S_1 \subset \cdots \subset S_r.$$

This can be read as a formal model of technical language formation.

An operation first appears as a compound construction:

$$m = t_m^A.$$

Repeated structural usefulness gives it high promotion gain.

The operation is then lexicalized as primitive:

$$t_m \rightsquigarrow m.$$

The transition preserves extensional meaning while reducing future derivational cost.

In this sense, primitive promotion is concept formation internal to a fixed algebraic theory.

The new symbol does not denote a new function. It creates a new direct access route to an existing function.

7.15 Promotion and Abbreviation

Ordinary mathematical abbreviation replaces a long expression by a shorter name. Primitive promotion is stronger because the abbreviation becomes available recursively inside all subsequent term constructions.

If

$$m = t_m$$

is introduced only as a textual abbreviation, every occurrence can be expanded back into t_m. If $m$is admitted as a primitive, exact minimum-size computation charges one operation node per use rather than the size of the expansion.

The gain may therefore multiply across nested and repeated use.

The promotion operation changes the cost model, not the extensional semantics.

This distinction is formalized by conservative definitional extension:

$$\Sigma \subseteq \Sigma' = \Sigma \cup \{m\},$$

together with the defining equation

$$m(x, y) \approx t_m(x, y).$$

The reduct of every Σ'-term back to Σpreserves its term function.

The extension changes complexity because expansion may increase size and height.

7.16 Reversibility

Every promotion can be reversed semantically by eliminating the promoted primitive using one of its defining terms.

Let

$$m = t_m^A$$

with $t_m$an S-term.

Define the elimination map

$$E_m : T_{S\cup\{m\}}(x, y) \to T_S(x, y)$$

recursively by replacing each occurrence of

$$m(r_1, r_2)$$

with

$$t_m(E_m(r_1), E_m(r_2)).$$

Then

$$E_m(t)^A = t^A.$$

Proposition 7.5. Primitive promotion is extensionally reversible.

Its complexity effect is generally not reversible without cost. Eliminating $m$may increase size by an amount depending on the number and nesting of occurrences.

Thus promotion is semantically conservative but metrically asymmetric.

7.17 Promotion Depth

Suppose $m$is promoted at stage k. Later promotions may depend heavily on m.

Define the promotion dependency relation

$$m_i \triangleleft m_j$$

when a minimum representation motivating promotion of m_j in S_j uses m_i.

The transitive closure produces a promotion dependency directed acyclic graph.

The depth of a promoted primitive is

$$d_{\mathrm{prom}}(m_j) = 1 + \max\{d_{\mathrm{prom}}(m_i) : m_i \triangleleft m_j\},$$

with depth one for operations promoted directly from the initial signature.

This records the hierarchical emergence of macros from earlier macros.

A recursive promotion path can therefore generate a layered vocabulary rather than an unstructured list of added operations.

7.18 Path Dependence

Let two promotion sequences begin from the same S_0:

$$S_0 \to S_1 \to \cdots \to S_r$$

and

$$S_0 \to T_1 \to \cdots \to T_r.$$

Even when

$$| S_r | = | T_r |,$$

the final signatures and costs may differ:

$$S_r \neq T_r,$$
$$T(S_r) \neq T(T_r).$$

The reason is that each promotion changes the marginal utility of all remaining candidates.

Definition 7.7. Promotion is path-independent at width k when every admissible greedy tie resolution produces the same terminal aggregate profile at width k.

Otherwise it is path-dependent.

The release freezes tie-breaking to ensure reproducibility. The frozen path is deterministic, but determinism does not imply path-independence as a mathematical property.

7.19 Cycle 9 Computation

Cycle 9 applied recursive promotion to the exact finite components obtained in the earlier cycles.

For each stage, the computation:

1. held the binary component fixed;
2. evaluated every unpromoted function as a candidate primitive;
3. recomputed exact minimum-size and minimum-height profiles;
4. ranked candidates by the declared gain criterion;
5. promoted the frozen winner;
6. recorded the new profile and manifest.

The resulting trajectories established that derived operations can be promoted recursively without changing binary content and that successive promotions can lower exact representation cost across the component.

The principal scientific content is not that additional primitives make terms shorter. That fact is immediate.

The substantive content is that the entire gain landscape is computable exactly:

$$m \longmapsto \Delta_T(S, m)$$

for every

$$m \in C \setminus S,$$

at every stage of the recursion.

Thus the next primitive is selected from a complete finite intervention space rather than by intuition.

7.20 Promotion Theorem

Theorem 7.3. Let (C, u)be a finite pointed binary component and $S_0 \in \mathcal{R}(C, u)$. Under any deterministic promotion policy, recursive primitive promotion produces a finite, reproducible, content-preserving trajectory

$$S_0 \subseteq S_1 \subseteq \cdots \subseteq S_r \subseteq C$$

such that

$$C_2(S_k; u) = C$$

for every k, while

$$T(S_{k+1}) \leq T(S_k)$$

and

$$H_{\max}(S_{k+1}) \leq H_{\max}(S_k).$$

If the selected candidate at stage $k$has positive gain, the corresponding inequality is strict for the declared objective.

The theorem establishes exact conservative reorganization of a born theory.

It does not establish that the greedy trajectory is globally optimal at every width.

7.21 Interpretation of Cycle 9

Cycle 9 extends Algebraic Abiogenesis from passive discovery to controlled intervention.

Earlier cycles ask:

What component has formed?
Which identities and structural certificates describe it?

Recursive promotion asks:

Which derived operation should become primitive next?

The answer is computed from the effect of each possible promotion on the entire exact component.

This yields the sequence

derived operation ⟶ measured global utility ⟶ primitive promotion ⟶ new access geometry.

Promotion is not spontaneous theory growth. The binary function set remains fixed.

It is endogenous language growth within a fixed theory.

The next section removes dependence on one promotion path by computing the full spectrum of singleton generators.

8. Full Single-Generator Birth Spectra

8.1 From Selected Orbits to Exhaustive Generator Sets

The earlier formation-profile calculations began from a distinguished binary primitive and examined operations obtained by input polarity reversal, output polarity reversal, and argument exchange. Because every operation in the resulting polarity orbit is interdefinable with the original primitive relative to the fixed involution u, the orbit supplies a natural family of content-preserving presentations.

That construction leaves a completeness question.

The polarity orbit may contain several singleton generators, but it need not contain every singleton generator of the pointed binary component. There may exist an operation

$$g \in C$$

outside the selected orbit such that

$$C_2(\{g\}; u) = C.$$

If so, an orbit-restricted formation spectrum would depend on the initially favored primitive. It would not describe the complete one-generator presentation space of the component.

Cycle 10 removes this dependence by testing every binary function in the component as a possible singleton generator.

Let (C, u)be a finite pointed binary term-function component.

Define

$$G(C,u) = \{g \in C : C_2(\{g\};u) = C\}.$$

This is the full singleton-generator set.

For every

$$g \in G(C,u),$$

compute the exact joint profile

$$\Psi(g) = \left(\mathbf{s}_g, \mathbf{h}_g, T(g), U(g), S_{\max}(g), H_{\max}(g)\right).$$

Definition 8.1. The full single-generator birth spectrum is the multiset

$$\mathfrak{G}(C,u) = [\Psi(g) : g \in G(C,u)],$$

where multiplicity is retained.

The multiset records not merely the distinct profile types but the number of generators realizing each type.

The adjective full means that every binary function in C is tested. It does not mean that higher-arity generators or complete finitary clones are included.

8.2 Exhaustive Generator Test

Let

$$N = |\ C\ |.$$

The full-generator census applies the following procedure to each

$$g \in C.$$

First, compute

$$C_g = C_2(\{g\};u).$$

Second, compare the resulting function-table set with the reference component:

$$C_g = C.$$

Third, if equality holds, compute the exact minimum-size and minimum-height profiles of all functions in C.

The generator test is exact because both C_g and C are finite sets of complete function tables.

Algorithm 8.1. Full single-generator spectrum

Input:

$$(C, u).$$

Initialize:

$$G = \emptyset.$$

For each

$$g \in C:$$

1. compute $C_g = C_2(\{g\}; u)$;
2. if $|\ C_g\ | \neq |\ C\ |$, reject g;
3. if $|\ C_g\ | = |\ C\ |$, compare the complete table sets;
4. if $C_g = C$, add g to G;
5. compute $\Psi(g)$.

Output:

$$G(C, u)$$

and

$$\mathfrak{G}(C, u).$$

The cardinality comparison in step two is an early rejection only. Equality of cardinality does not replace equality of function sets.

Proposition 8.1. Algorithm 8.1 returns every singleton generator of (C, u).

Proof. Every singleton generator belongs to C, because

$$g \in C_2(\{g\}; u) = C.$$

The algorithm tests every $g \in C$. It accepts precisely those satisfying the defining condition

$$C_2(\{g\}; u) = C.$$

Therefore no singleton generator is omitted and no nongenerator is accepted. □

The computation is exhaustive within the declared binary component.

8.3 Generator Density

Define the singleton-generator density by

$$d_1(C, u) = \frac{| G(C, u) |}{| C |}.$$

This is the probability that a uniformly selected binary function from C singly regenerates the whole component with the distinguished unary operation.

The exact densities in the three principal families are

$$d_1(C_B, u_B) = \frac{8}{16} = \frac{1}{2},$$
$$d_1(C_W, u_W) = \frac{8}{24} = \frac{1}{3},$$

and

$$d_1(C_S, u_S) = \frac{8}{82} = \frac{4}{41}.$$

The Strong Kleene component contains substantially more binary functions than the other two, while retaining only eight singleton generators.

The density values are descriptive invariants of the tested pointed components. No general monotonic law relating component size to generator density is inferred from three examples.

8.4 Exact Generator Counts

The exhaustive scans tested:

$$16$$

Boolean binary functions,

$$24$$

Weak Kleene binary functions, and

$$82$$

Strong Kleene binary functions.

Thus the total number of singleton candidates tested was

$$16 + 24 + 82 = 122.$$

In each family, exactly eight candidates regenerated the complete component.

Family	$\mid C \mid$	$\mid G(C,u) \mid$	$d_1(C,u)$
Boolean meet-negation	16	8	$1/2$
Weak Kleene	24	8	$1/3$
Strong Kleene	82	8	$4/41$

Theorem 8.1. For each of the Boolean meet-negation, Weak Kleene, and Strong Kleene pointed binary components,

$$\mid G(C,u) \mid = 8.$$

Across the three exhaustive scans, twenty-four of the 122 tested binary functions are full singleton generators.

The equality of generator counts does not imply equality of generator spectra. The generated components, profile costs, and complexity distributions remain different.

8.5 Equality with the Polarity Orbit

Let $b$be the original binary primitive used to generate one of the three reference components. Let

$$\mathrm{Orb}_u(b)$$

be its polarity-permutation orbit.

Earlier cycles established

$$\mathrm{Orb}_u(b) \subseteq G(C,u),$$

because every orbit member is interdefinable with $b$relative to u.

The exhaustive scan establishes the reverse inclusion.

$$G(C,u) \subseteq \mathrm{Orb}_u(b).$$

Therefore

$$G(C,u) = \mathrm{Orb}_u(b).$$

Theorem 8.2. In each of the three principal pointed components,

$$G(C,u) = \mathrm{Orb}_u(b).$$

Thus the polarity orbit contains every binary operation that singly regenerates the complete binary component with the distinguished involution.

Proof. Every orbit operation generates C, giving

$$\mathrm{Orb}_u(b) \subseteq G(C,u).$$

The exhaustive census finds exactly eight singleton generators. The polarity orbit also contains exactly eight distinct operations in each case. Hence the finite inclusion is an equality. □

This result upgrades the earlier orbit study from a selected presentation family to the complete singleton-generator census.

The agreement was not assumed. It was obtained by testing every member of each component.

8.6 Orbit Sufficiency

The equality

$$G(C,u) = \mathrm{Orb}_u(b)$$

has two consequences.

First, no hidden singleton basis exists outside the polarity transformations of the original primitive.

Second, every one-generator formation profile previously observed in the orbit analysis belongs to the complete singleton spectrum.

Thus the Cycle 7 profile ranges were not artifacts of an arbitrarily chosen local orbit.

They exhaust all singleton presentations of the three principal pointed components.

Corollary 8.1. For the Boolean, Weak Kleene, and Strong Kleene components,

$$\mathfrak{F}_1(C,u) = \mathfrak{F}_{\mathrm{orb}}(b;u).$$

The complete one-generator spectrum is exactly the polarity-orbit spectrum.

This conclusion is specific to the three exhaustively scanned components. It is not asserted for arbitrary finite pointed components.

8.7 Profile Equivalence Classes

Define an equivalence relation on $G(C,u)$by

$$g \sim_\Psi h \Leftrightarrow \Psi(g) = \Psi(h).$$

Equivalently,

$$s_g(f) = s_h(f)$$

and

$$h_g(f) = h_h(f)$$

for every canonically identified function $f \in C$, after accounting for the symmetry used in normalized profile comparison.

The quotient

$$G(C,u)/\sim_\Psi$$

is the set of distinct exact profile types.

In all three principal families,

$$| G(C,u)/\sim_{\Psi} | = 3.$$

The eight generators divide into profile-class multiplicities

$$2,\ 4,\ 2.$$

Theorem 8.3. Each principal full singleton-generator spectrum contains exactly three distinct joint size-height profile types, with multiplicities

$$2 + 4 + 2 = 8.$$

The same multiplicity pattern occurs in the Boolean, Weak Kleene, and Strong Kleene components.

The equality of multiplicity patterns does not imply equality of the underlying profiles. The exact layer counts and aggregate costs differ by family.

8.8 Boolean Full-Generator Spectrum

For the Boolean component,

$$| C_B | = 16.$$

The eight singleton generators separate into three profile classes.

Type	Multiplicity	T	U	$S_{\max}$	$H_{\max}$
B_1	2	37	31	5	3
B_2	4	30	28	5	4
B_3	2	41	33	7	4

The exact size-layer counts are

$$N^s(B_1) = (2,3,4,4,1,2),$$
$$N^s(B_2) = (2,5,5,2,1,1),$$

and

$$N^s(B_3) = (2,3,4,4,1,0,0,2).$$

The exact height-layer counts are

$$N^h(B_1) = (2,3,5,6),$$
$$N^h(B_2) = (2,5,5,3,1),$$

and

$$N^h(B_3) = (2,3,5,4,2).$$

Type B_2minimizes total size:

$$T(B_2) = 30.$$

It also minimizes total height:

$$U(B_2) = 28.$$

However,

$$H_{\max}(B_2) = 4,$$

whereas

$$H_{\max}(B_1) = 3.$$

Thus the total-cost optimum does not minimize saturation height.

Proposition 8.2. The Boolean singleton spectrum has no single profile type minimizing all four aggregate criteria

$$T,\ U,\ S_{\max},\ H_{\max}.$$

Proof. Type B_2minimizes $T$and $U$and ties for minimum $S_{\max}$, while type B_1uniquely attains the lower saturation height. □

The Boolean result gives an exact multiobjective tradeoff inside the complete set of singleton bases.

8.9 Weak Kleene Full-Generator Spectrum

For the Weak Kleene component,

$$| C_W |= 24.$$

The three profile classes are

Type	Multiplicity	T	U	$S_{\max}$	$H_{\max}$
W_1	2	63	52	5	3
W_2	4	48	44	5	4
W_3	2	69	57	7	4

The exact size-layer counts are

$$N^s(W_1) = (2,3,5,9,2,3),$$
$$N^s(W_2) = (2,6,9,5,1,1),$$

and

$$N^s(W_3) = (2,3,5,6,6,0,0,2).$$

The height-layer counts are

$$N^h(W_1) = (2,3,8,11),$$
$$N^h(W_2) = (2,6,11,4,1),$$

and

$$N^h(W_3) = (2,3,7,8,4).$$

Type W_2 minimizes total size:

$$T(W_2) = 48,$$

and total height:

$$U(W_2) = 44.$$

Type W_1 attains the lower saturation height:

$$H_{\max}(W_1) = 3.$$

Both W_1 and W_2 attain saturation size five.

The Weak Kleene spectrum therefore has the same aggregate tradeoff pattern as the Boolean spectrum:

$$\text{best total cost} \neq \text{best maximum height.}$$

The numerical costs differ because the component contains twenty-four rather than sixteen functions.

8.10 Strong Kleene Full-Generator Spectrum

For the Strong Kleene component,

$$| C_S | = 82.$$

The full spectrum contains:

Type	Multiplicity	T	U	$S_{\max}$	$H_{\max}$
S_1	2	368	266	8	5
S_2	4	291	215	6	4
S_3	2	444	302	10	5

The exact size-layer counts are

$$N^s(S_1) = (2,3,7,11,14,20,15,8,2),$$
$$N^s(S_2) = (2,6,12,20,19,13,10),$$

and

$$N^s(S_3) = (2,3,5,9,12,12,11,9,6,11,2).$$

The height-layer counts are

$$N^h(S_1) = (2,3,8,37,24,8),$$
$$N^h(S_2) = (2,6,20,47,7),$$

and

$$N^h(S_3) = (2,3,8,13,36,20).$$

Type S_2strictly improves every aggregate criterion relative to both other types:

$$291 < 368 < 444,$$
$$215 < 266 < 302,$$
$$6 < 8 < 10,$$

and

$$4 < 5.$$

Proposition 8.3. The Strong Kleene singleton spectrum contains a unique aggregate profile type that minimizes

$$T,\ U,\ S_{\max},\ H_{\max}$$

simultaneously.

This optimum has multiplicity four.

The result does not mean that its four generators have identical operation tables. They are distinct primitive functions sharing the same exact formation profile.

8.11 Profile Multiplicity and Symmetry

The repeated multiplicity pattern

$$2,\ 4,\ 2$$

reflects symmetry inside the polarity orbit.

Operations related by transformations preserving the normalized formation profile fall into the same $\sim_{\Psi}$-class.

However, polarity interdefinability alone does not force profile equality. All eight generators belong to one interdefinability orbit, yet they split into three cost types.

Thus there are two distinct quotient structures:

$$G(C,u)/\sim_{\mathrm{orb}}$$

and

$$G(C,u)/\sim_{\Psi}.$$

For the principal examples,

$$G(C,u)/\sim_{\mathrm{orb}}$$

has one class, because all generators occupy one polarity orbit.

By contrast,

$$G(C,u)/\sim_{\Psi}$$

has three classes.

Therefore

$$\text{interdefinability symmetry} \Rightarrow/\text{cost symmetry}.$$

The complexity metric refines the orbit relation.

8.12 Complete Singleton Pareto Sets

For singleton generators, define aggregate dominance by

$$g \preceq_{\mathrm{agg}} h$$

when

$$\begin{aligned} &T(g) \le T(h),\\ &U(g) \le U(h),\\ &S_{\max}(g) \le S_{\max}(h), \end{aligned}$$

and

$$H_{\max}(g) \le H_{\max}(h).$$

Strict dominance requires at least one strict inequality.

The aggregate Pareto set is

$$\mathrm{Par}_1(C,u) = \{g \in G(C,u) : \text{no } h \in G(C,u) \text{ strictly dominates } g\}.$$

For the Boolean and Weak Kleene components, the low-total-cost type and the low-height type are both Pareto-relevant.

The high-cost type is dominated.

For Strong Kleene, the central type dominates both alternatives in all reported aggregate coordinates.

Thus the singleton optimization geometry differs qualitatively:

Boolean and Weak Kleene: tradeoff,
Strong Kleene: aggregate dominance.

The complete generator scan is required to establish that no omitted singleton primitive lies outside these Pareto classifications.

8.13 Generator Spectrum as an Isomorphism Invariant

Let

$$\varphi: A \to B$$

be an isomorphism between finite pointed binary components

$$(C_A, u_A)$$

and

$$(C_B, u_B).$$

For every binary function $f \in C_A$, define

$$\varphi_*(f)(x,y) = \varphi(f(\varphi^{-1}(x), \varphi^{-1}(y))).$$

The map

$$\varphi_*: C_A \to C_B$$

is a bijection preserving projections, unary action, and binary superposition.

If

$$g \in G(C_A, u_A),$$

then

$$\varphi_*(g) \in G(C_B, u_B).$$

The syntax of every representing term is unchanged under transport.

Therefore

$$s_g(f) = s_{\varphi_*(g)}(\varphi_*(f))$$

and

$$h_g(f) = h_{\varphi_*(g)}(\varphi_*(f)).$$

Theorem 8.4. The full single-generator birth spectrum is invariant under isomorphism of finite pointed binary components.

More precisely,

$$\mathfrak{G}(C_A, u_A) = \mathfrak{G}(C_B, u_B)$$

as multisets of normalized exact profiles.

The spectrum is therefore not an artifact of carrier labeling.

8.14 Spectrum Hashes and Reproducibility

For computational preservation, each exact pointwise profile may be serialized canonically and assigned a cryptographic digest.

Let

$$\mathrm{ser}(\Psi(g))$$

be the normalized serialization of:

$$\mathrm{tab}(f), s_g(f), h_g(f)$$

for every canonically ordered $f \in C$.

Define

$$H(g) = \mathrm{SHA256}(\mathrm{ser}(\Psi(g))).$$

The full spectrum hash is obtained from the sorted multiset of individual profile hashes.

This allows independent recomputation to verify:

generator count,
profile count,
profile multiplicities,

and

complete pointwise costs.

Aggregate agreement alone is insufficient. Two different profiles may share the same totals and saturation radii.

The normalized digest certifies the complete profile rather than only its summary statistics.

8.15 Completeness Relative to Arity Two

The full-generator spectrum remains arity-bounded.

The condition

$$g \in G(C, u)$$

means

$$C_2(\{g\}; u) = C.$$

It does not establish equality of all n-ary term-function components:

$$C_n(\{g\}; u)$$

for every $n \geq 1$.

Accordingly, the spectrum is a complete census of singleton generators of the binary component, not necessarily a complete census of singleton generators of the full finitary clone.

The exact result is:

all binary functions in C were tested as binary-component generators.

The stronger claim

$$\mathrm{Clo}(u, g) = \mathrm{Clo}(u, b)$$

requires independent higher-arity analysis.

8.16 Scope of the Larger Components

The exhaustive singleton scan was completed for the components of sizes

$$16,\ 24,\ 82.$$

The larger BCK-related components of sizes

$$1152$$

and

$$2688$$

were not exhaustively scanned over every possible singleton binary primitive.

Their Cycle 7–9 profiles remain exact for the tested presentation families, but they are not complete full-generator spectra.

Thus the statement

$$G(C, u) = \mathrm{Orb}_u(b)$$

is established only for the three principal components.

No analogous equality is claimed for the larger components.

This boundary is computational rather than conceptual. The same definition and algorithm apply, but the cost of evaluating every candidate generator and its full exact profile increases substantially.

8.17 Discovery Significance

The full-generator census answers a question that recursive promotion alone cannot answer.

Recursive promotion begins from one presentation and follows one intervention path.

The full singleton spectrum asks instead:

What are all possible one-binary-primitive origins of this component?

For the three principal systems, the answer is unexpectedly compact:

8 generators,
3 exact profile types,

and

1 polarity orbit.

The component does not admit a hidden isolated singleton basis with a radically different profile.

The exact one-generator presentation ecology is exhausted by polarity transformations of the original primitive.

This result makes the singleton landscape closed rather than exploratory.

8.18 Cycle 10 Theorem

Theorem 8.5. Full Single-Generator Spectrum Theorem.
For each of the Boolean meet-negation, Weak Kleene, and Strong Kleene pointed binary term-function components:

1. every binary function in the component was tested as a singleton generator;
2. exactly eight functions regenerate the complete binary component with the distinguished involution;
3. the eight generators are exactly the polarity-permutation orbit of the original primitive;

4. the eight generators divide into exactly three joint minimum-size and minimum-height profile classes;
5. the profile-class multiplicities are $2,4,2$;
6. the resulting multiset of profiles is invariant under pointed-component isomorphism.

The theorem upgrades a presentation-dependent orbit calculation into a complete singleton-generator invariant for the three finite components.

It also fixes the baseline for width-two analysis.

Once every singleton generator is known, every preserving pair can be classified as either:

a singleton generator plus another primitive,

or

an irreducibly cooperative pair.

The next section exhausts that width-two space.

9. Exact Signature-Refactoring Landscapes

9.1 Width-Two Presentation Space

Let (C, u)be a finite pointed binary term-function component. The full singleton-generator family is

$$\mathcal{R}_1(C,u) = \{\{g\} \subseteq C : C_2(\{g\}; u) = C\}.$$

The preserving pair family is

$$\mathcal{R}_2(C,u) = \{\{g,h\} \subseteq C : g \neq h,\ C_2(\{g,h\}; u) = C\}.$$

The exact width-two signature space is

$$\mathcal{R}_{\leq 2}(C,u) = \mathcal{R}_1(C,u) \cup \mathcal{R}_2(C,u).$$

Every unordered pair of distinct binary functions in $C$is tested. Hence the number of pair candidates is

$$\binom{|\,C\,|}{2}.$$

The three principal components give

$$\binom{16}{2} = 120,$$
$$\binom{24}{2} = 276,$$

and

$$\binom{82}{2} = 3321.$$

Thus the total exhaustive pair census contains

$$120 + 276 + 3321 = 3717$$

candidate signatures.

No sampling or evolutionary search is used at this stage. The width-two landscape is complete relative to the fixed binary component and fixed unary involution.

9.2 Signature States

A state is a preserving binary signature

$$S \in \mathcal{R}_{\leq 2}(C, u).$$

Each state is labeled by the exact descriptor

$$\Phi(S) = (|\,S\,|, T(S), S_{\max}(S), H_{\max}(S)),$$

where

$$T(S) = \sum_{f \in C} s_S(f),$$
$$S_{\max}(S) = \max_{f \in C} s_S(f),$$

and

$$H_{\max}(S) = \max_{f \in C} h_S(f).$$

The complete pointwise record also contains

$$\mathbf{s}_S = (s_S(f))_{f \in C}$$

and

$$\mathbf{h}_S = (h_S(f))_{f \in C}.$$

Two states may have equal aggregate descriptors while possessing different pointwise profiles. The repository therefore retains a normalized digest of the complete joint profile.

The state space holds extensional content fixed:

$$C_2(S; u) = C$$

for every retained state.

Only the primitive inventory and resulting access geometry vary.

9.3 Promotion and Demotion Edges

Let

$$S = \{g\} \in \mathcal{R}_1(C, u)$$

and let

$$S^+ = \{g, h\} \in \mathcal{R}_2(C, u).$$

A promotion edge is

$$\{g\} \longrightarrow \{g, h\}.$$

The reverse transition is a demotion edge:

$$\{g, h\} \longrightarrow \{g\}.$$

A demotion is retained only when the remaining singleton still generates C.

The resulting directed graph is

$$\mathbf{R}_2(C,u) = (\mathcal{R}_{\leq 2}(C,u), E).$$

Every edge preserves the binary component.

The graph is bipartite by width. Promotion edges move from width one to width two. Demotion edges move from width two to width one.

No direct pair-to-pair replacement edge is required. Replacing $g$by $h$may be represented as

$$\{g,m\} \longrightarrow \{m\} \longrightarrow \{h,m\},$$

provided that the intermediate singleton $\{m\}$preserves the component.

9.4 Edge Weights

A promotion has both a construction cost and an access effect.

If $h$is promoted from singleton state $\{g\}$, define its promotion-definition cost by

$$c_{\mathrm{prom}}(\{g\},h) = s_{\{g\}}(h).$$

This is the exact minimum number of operation nodes required to define $h$before it becomes primitive.

The change in global access cost is

$$\delta_T(S,S') = T(S') - T(S).$$

A negative value indicates improvement.

The changes in saturation radii are

$$\delta_{S_{\max}}(S,S') = S_{\max}(S') - S_{\max}(S)$$

and

$$\delta_{H_{\max}}(S,S') = H_{\max}(S') - H_{\max}(S).$$

A promotion may have positive acquisition cost while producing a negative access-cost change.

Demotion is assigned zero acquisition cost. It may nevertheless increase subsequent representation costs because one primitive is removed.

Each directed edge therefore records

$$\left(c_{\mathrm{prom}}, \delta_T, \delta_{S_{\max}}, \delta_{H_{\max}}\right).$$

9.5 Exact Landscape Census

The exhaustive counts are

Component	$\lvert C \rvert$	Pairs tested	$\lvert \mathcal{R}_2 \rvert$	Cooperative pairs
Boolean	16	120	92	0
Weak Kleene	24	276	156	0
Strong Kleene	82	3321	620	0

The total number of preserving pairs is

$$92 + 156 + 620 = 868.$$

The aggregate preserving-pair density is therefore

$$\frac{868}{3717} \approx 0.2335.$$

The family-specific densities are

$$\frac{92}{120} = \frac{23}{30} \approx 0.7667,$$
$$\frac{156}{276} = \frac{13}{23} \approx 0.5652,$$

and

$$\frac{620}{3321} \approx 0.1867.$$

Preserving pairs become proportionally less common across these three increasingly large components.

No universal monotonicity theorem is inferred from the three examples.

9.6 Graph Sizes

Each principal component has eight preserving singleton signatures.

Hence the graph-node counts are

$$8 + 92 = 100$$

for Boolean,

$$8 + 156 = 164$$

for Weak Kleene, and

$$8 + 620 = 628$$

for Strong Kleene.

The directed edge counts are

$$240{,}368{,}1296,$$

respectively.

Thus the three exact landscapes contain

$$100 + 164 + 628 = 892$$

states and

$$240 + 368 + 1296 = 1904$$

directed promotion or demotion edges.

Every retained state is reachable from the original signature within the width-two graph.

Component	Graph nodes	Directed edges	Nodes reachable from original
Boolean	100	240	100
Weak Kleene	164	368	164
Strong Kleene	628	1296	628

The reachable-state equality

$$| \operatorname{Reach}(S_0) | = | \mathcal{R}_{\leq 2}(C, u) |$$

means that the original presentation belongs to the same preserving refactoring component as every singleton and pair state found in the exhaustive census.

No content-changing excursion is required to reach any retained width-two presentation.

9.7 Original, Singleton, Anchored-Pair, and Global-Pair Optima

Let

$$S_0 = \{b_0\}$$

be the original binary primitive signature.

Let

$$S_1^* \in \operatorname*{arg\,min}_{S \in \mathcal{R}_1(C,u)} T(S)$$

be a best singleton signature.

Let

$$S_{2,0}^* \in \arg\min\{T(S) : S \in \mathcal{R}_2(C, u),\ b_0 \in S\}$$

be a best original-anchored pair.

Let

$$S_2^* \in \operatorname*{arg\,min}_{S \in \mathcal{R}_2(C,u)} T(S)$$

be a globally best pair.

The exact values are

Component	$T(S_0)$	$T(S_1^*)$	$T(S_{2,0}^*)$	$T(S_2^*)$
Boolean	41	30	25	22
Weak Kleene	69	48	44	36
Strong Kleene	444	291	223	190

The corresponding saturation pairs are

Component	S_0	S_1^*	$S_{2,0}^*$	S_2^*
Boolean	$7/4$	$5/4$	$3/2$	$3/2$
Weak Kleene	$7/4$	$5/4$	$3/2$	$3/2$
Strong Kleene	$10/5$	$6/4$	$5/4$	$5/3$

Here

$$a/b$$

denotes

$$S_{\max} = a, H_{\max} = b.$$

The best global pair need not contain the original primitive.

Indeed,

$$T(S_2^*) < T(S_{2,0}^*)$$

in all three systems.

The improvements beyond the best original-anchored pair are

$$25 - 22 = 3,$$
$$44 - 36 = 8,$$

and

$$223 - 190 = 33.$$

Thus direct promotion from the original signature does not reach the globally best width-two presentation. Primitive replacement is beneficial before the final macro is selected.

9.8 Boolean Landscape

The original Boolean presentation has

$$T(S_0) = 41,$$
$$S_{\max}(S_0) = 7,$$

and

$$H_{\max}(S_0) = 4.$$

The best singleton lowers total size by

$$41 - 30 = 11.$$

The best pair lowers total size by

$$41 - 22 = 19.$$

The relative reduction is

$$\frac{19}{41} \approx 0.4634.$$

The saturation size decreases from

$$7$$

to

$$3,$$

and saturation height decreases from

$$4$$

to

$$2.$$

The best original-anchored pair has total cost

$$25.$$

It captures most, but not all, of the available width-two improvement.

The globally best pair yields an additional reduction of three operation nodes in aggregate.

The Boolean landscape contains

$$100$$

states and

$$240$$

directed edges. Every state is reachable from the original meet-based presentation through preserving promotions and demotions.

9.9 Weak Kleene Landscape

The original Weak Kleene presentation has

$$T(S_0) = 69,$$
$$S_{\max}(S_0) = 7,$$

and

$$H_{\max}(S_0) = 4.$$

The best singleton gives

$$T(S_1^*) = 48.$$

The best pair gives

$$T(S_2^*) = 36.$$

The exact reduction from the original is

$$69 - 36 = 33,$$

with relative reduction

$$\frac{33}{69} = \frac{11}{23} \approx 0.4783.$$

The saturation pair improves from

$$7/4$$

to

$$3/2.$$

The best pair containing the original primitive has cost

$$44.$$

Replacing the original primitive before adding the final macro produces an additional saving of

$$44 - 36 = 8.$$

The exact graph contains

$$164$$

states and

$$368$$

directed edges, all reachable from the original state.

9.10 Strong Kleene Landscape

The Strong Kleene component produces the largest exact landscape.

The original presentation has

$$T(S_0) = 444,$$
$$S_{\max}(S_0) = 10,$$

and

$$H_{\max}(S_0) = 5.$$

The best singleton reduces total size to

$$291.$$

The best original-anchored pair reduces it further to

$$223.$$

The globally best pair gives

$$T(S_2^*) = 190.$$

The total reduction from the original is

$$444 - 190 = 254.$$

The relative reduction is

$$\frac{254}{444} = \frac{127}{222} \approx 0.5721.$$

The saturation size falls from

$$10$$

to

$$5,$$

and the saturation height falls from

$$5$$

to

$$3.$$

The gain beyond the best original-anchored pair is

$$223 - 190 = 33.$$

Thus the largest component also displays the strongest absolute and relative benefit from global signature replacement.

The exact Strong Kleene landscape contains

$$628$$

states and

$$1296$$

directed edges. Every state is reachable from the original primitive under the preserving width-two transition policy.

9.11 Aggregate Reduction Theorem

Theorem 9.1. In each principal component, the globally best preserving pair strictly improves the original signature in total minimum operation-node cost, saturation size, and saturation height.

The exact reductions are

Component	ΔT	$\Delta S_{\max}$	$\Delta H_{\max}$
Boolean	19	4	2
Weak Kleene	33	4	2
Strong Kleene	254	5	2

where

$$\Delta T = T(S_0) - T(S_2^*),$$
$$\Delta S_{\max} = S_{\max}(S_0) - S_{\max}(S_2^*),$$

and

$$\Delta H_{\max} = H_{\max}(S_0) - H_{\max}(S_2^*).$$

The theorem establishes exact width-two compression.

It does not establish unconditional superiority of pair signatures because primitive width has not yet been charged.

9.12 Pareto Structure

Define aggregate dominance on preserving signatures by

$$S \preceq_{\mathrm{P}} T$$

when

$$|S| \leq |T|,$$
$$T(S) \leq T(T),$$
$$S_{\max}(S) \leq S_{\max}(T),$$

and

$$H_{\max}(S) \leq H_{\max}(T),$$

with at least one strict inequality for strict dominance.

The aggregate Pareto frontier consists of undominated states.

For Boolean, the frontier contains three aggregate types:

$$(1,30,5,4),$$
$$(1,37,5,3),$$

and

$$(2,22,3,2),$$

with multiplicities

$$4,\ 2,\ 8.$$

For Weak Kleene, the frontier contains

$$(1,48,5,4),$$
$$(1,63,5,3),$$

and

$$(2,36,3,2),$$

with multiplicities

$$4,\ 2,\ 4.$$

For Strong Kleene, the frontier contains

$$(1{,}291{,}6{,}4),$$
$$(2{,}190{,}5{,}3),$$

and

$$(2{,}193{,}4{,}3),$$

with multiplicities

$$4,\ 8,\ 8.$$

The Strong Kleene pair minimizing total cost does not minimize saturation size.

The total-cost optimum has

$$T = 190, S_{\max} = 5, H_{\max} = 3.$$

A distinct pair type has

$$T = 193, S_{\max} = 4, H_{\max} = 3.$$

Thus three additional aggregate operation nodes reduce the worst-case minimum size by one.

This is an exact multiobjective tradeoff:

$$\text{minimum total work} \neq \text{minimum worst-case size}.$$

9.13 Shortest Preserving Refactoring Paths

A path

$$P = (S_0, S_1, \ldots, S_r)$$

in the refactoring graph is assigned cumulative promotion-definition size

$$C_{\mathrm{prom}}(P) = \sum_{\substack{0 \le i < r \\ S_i \to S_{i+1} \text{ promotion}}} c_{\mathrm{prom}}\,(S_i, S_{i+1}).$$

Demotions contribute zero acquisition cost.

The path comparison is lexicographic:

$$\big(C_{\mathrm{prom}}(P), r\big).$$

The first objective minimizes cumulative definition size. The second minimizes transition count among equal-cost paths.

For each principal component, a shortest path from the original signature to a globally best pair has three transitions:

$$\{b_0\} \longrightarrow \{b_0, g^*\} \longrightarrow \{g^*\} \longrightarrow \{g^*, m^*\}.$$

The first transition promotes a replacement singleton generator.

The second removes the original primitive.

The third promotes the final macro.

The exact cumulative promotion-definition sizes are

Component	C_{prom}	Transitions
Boolean	6	3
Weak Kleene	4	3
Strong Kleene	8	3

The path remains inside the preserving-signature graph at every stage.

No temporary enlargement or loss of the binary component occurs.

9.14 Stepwise Access Changes

The Boolean shortest path changes total cost by

$$-13, +2, -8.$$

The net change is

$$-13 + 2 - 8 = -19.$$

The temporary demotion increases access cost by two, but enables transition from the original singleton to the better singleton basis.

The Weak Kleene path changes total cost by

$$-23, +2, -12,$$

with net change

$$-33.$$

The Strong Kleene path changes total cost by

$$-171, +18, -101,$$

with net change

$$-254.$$

Thus a globally optimal refactoring path need not improve the access objective monotonically at every edge.

A demotion may temporarily raise

$$T(S)$$

while remaining necessary for reaching a superior final signature within the width bound.

Proposition 9.1. Endpoint-optimal preserving refactoring need not be edgewise access-monotone.

Proof. Each recorded shortest path contains a demotion edge with positive access-cost change:

$$+2, +2, +18.$$

Nevertheless, each path ends at the global minimum pair cost. □

This distinguishes path optimization from greedy one-step compression.

9.15 Direct Anchored Promotion Versus Global Refactoring

The best original-anchored pair is reachable in one promotion:

$$\{b_0\} \longrightarrow \{b_0, m_0^*\}.$$

The exact promotion-definition sizes are

$$7,7,6$$

for Boolean, Weak Kleene, and Strong Kleene.

These one-step paths produce final total costs

$$25,44,223.$$

The globally best pair requires three transitions but gives lower costs:

$$22,36,190.$$

Thus there are two distinct notions of inexpensive change.

The first minimizes transition count from the existing primitive inventory.

The second minimizes the final access objective after allowing replacement of the original primitive.

A direct macro addition is operationally simple but globally suboptimal.

A basis replacement followed by macro promotion is structurally more expensive but yields a better final presentation.

9.16 Refactoring Distance

Define the unweighted transition distance

$$d_{\mathrm{tr}}(S, T)$$

as the minimum number of promotion or demotion edges between preserving signatures $S$and T.

Define the acquisition distance

$$d_{\mathrm{acq}}(S,T)$$

as the minimum cumulative promotion-definition size over all preserving paths.

These distances need not select the same path.

The direct anchored pair has

$$d_{\mathrm{tr}}(S_0, S_{2,0}^{*}) = 1.$$

The globally best pair has a recorded shortest path with

$$d_{\mathrm{tr}}(S_0, S_2^{*}) = 3.$$

Its acquisition distances are

$$6,\ 4,\ 8.$$

The landscape therefore supports at least three optimization layers:

final access cost,
transition distance,

and

acquisition distance.

The best endpoint depends on how these quantities are weighted.

9.17 Landscape Connectedness

Because all retained states are reachable from the original state, the underlying undirected graph of each principal landscape is connected.

Theorem 9.2. The exact width-two preserving-signature graphs for Boolean meet-negation, Weak Kleene, and Strong Kleene are connected.

More strongly, every state is reachable from the original signature in the directed graph containing both promotion and demotion transitions.

This result depends on the actual finite landscapes.

Preserving-signature graphs need not be connected in general. A component could possess two minimal generating signatures that cannot be joined under a strict width bound without passing through a nonpreserving intermediate state.

The observed connectedness therefore constitutes additional finite structure of the three examples.

9.18 Exactness and Reproducibility

For every state, the release records:

primitive operation tables,
component-preservation decision,
exact size profile,
exact height profile,
$T(S)$,
$S_{\max}(S)$,
$H_{\max}(S)$,

and

normalized profile digest.

For every edge, it records:

source,
target,
promotion or demotion action,
added or removed primitive,
promotion-definition cost,

and

changes in access statistics.

The complete state and edge tables determine the finite landscape independently of graphical visualization.

Each family also receives a landscape-level cryptographic digest.

The exact object reported by Cycle 11 is therefore not merely a best score. It is the complete weighted width-two preserving-signature graph.

9.19 Exact Signature-Landscape Theorem

Theorem 9.3. For the Boolean meet-negation, Weak Kleene, and Strong Kleene pointed binary components, Cycle 11 exhaustively constructs the full width-two preserving-signature landscape.

The construction establishes:

1. all 3717unordered pair signatures are tested;
2. exactly 868pairs preserve their corresponding binary components;
3. the three preserving graphs contain 892states and 1904directed edges;
4. every retained state is reachable from the original signature;
5. the best pair totals are 22, 36, and 190;
6. the best-pair saturation size-height pairs are $3/2$, $3/2$, and $5/3$;
7. each globally best pair is reachable through a three-transition component-preserving path;
8. the cumulative promotion-definition sizes of those paths are 6, 4, and 8;
9. global pair optimization may require temporary increase of access cost during primitive replacement;
10. the Strong Kleene landscape contains a strict tradeoff between minimum total size and minimum saturation size.

The theorem identifies a new finite object attached to a born algebraic component:

the exact weighted ecology of its bounded primitive presentations.

The next section determines the generative status of the preserving pairs. In particular, it asks whether any pair achieves completeness only through interaction between two individually incomplete primitives.

10. Cooperative Generation and Its Absence

10.1 Two Mechanisms of Pair Completeness

Let (C, u)be a finite pointed binary term-function component, and let

$$\{g, h\} \subseteq C$$

be a preserving pair:

$$C_2(\{g, h\}; u) = C.$$

There are two logically distinct mechanisms by which this equality can occur.

The first is inherited completeness. At least one member already generates the whole component:

$$C_2(\{g\}; u) = C$$

or

$$C_2(\{h\}; u) = C.$$

The pair then consists of a complete singleton basis together with an additional primitive. The second primitive may reduce representation cost, but it contributes no new binary content.

The second mechanism is cooperative completeness. Neither member is complete alone:

$$C_2(\{g\}; u) \neq C,$$
$$C_2(\{h\}; u) \neq C,$$

while jointly

$$C_2(\{g, h\}; u) = C.$$

In the cooperative case, completeness is an emergent property of the pair. It cannot be attributed to either primitive separately.

These mechanisms must not be conflated.

A preserving pair of the first kind is a refactored complete basis.

A preserving pair of the second kind is a minimal width-two generator.

Definition 10.1. A preserving pair $\{g, h\}$is inherited when

$$g \in G(C, u)$$

or

$$h \in G(C, u).$$

It is cooperative when

$$g \notin G(C, u), h \notin G(C, u),$$

but

$$C_2(\{g, h\}; u) = C.$$

The two classes partition the preserving pair family:

$$\mathcal{R}_2(C, u) = \mathcal{I}_2(C, u) \sqcup \mathcal{K}_2(C, u),$$

where $\mathcal{I}_2$denotes inherited pairs and $\mathcal{K}_2$denotes cooperative pairs.

10.2 Minimal Generating Signatures

Recall that a preserving signature $S$is inclusion-minimal when no proper subset of Spreserves C.

For a pair,

$$S = \{g, h\},$$

the proper nonempty subsets are

$$\{g\}\text{and}\{h\}.$$

Therefore:

$$\{g, h\} \text{ is inclusion-minimal}$$

if and only if

$$C_2(\{g\}; u) \neq C,$$
$$C_2(\{h\}; u) \neq C,$$

and

$$C_2(\{g, h\}; u) = C.$$

Proposition 10.1. A preserving pair is cooperative if and only if it is an inclusion-minimal generating signature of width two.

Proof. The defining conditions are identical. □

Thus the cooperative-pair search is exactly the search for new minimal generators beyond the known singleton bases.

The complete minimal-generator family satisfies

$$\mathrm{MinGen}(C, u) = \mathcal{R}_1(C, u) \cup \mathcal{K}_2(C, u) \cup \mathrm{MinGen}_{\geq 3}(C, u).$$

Cycle 11 determines the first two terms. It does not determine whether minimal generating signatures of width three or greater exist.

10.3 Lattice Formulation

For $g \in C$, let

$$C_g = C_2(\{g\}; u)$$

be the singleton-generated subcomponent.

The join of two such subcomponents is

$$C_g \vee C_h = C_2(C_g \cup C_h; u).$$

Because $g \in C_g$ and $h \in C_h$,

$$C_g \vee C_h = C_2(\{g, h\}; u).$$

A cooperative pair therefore satisfies

$$C_g < C,$$

$$C_h < C,$$

but

$$C_g \vee C_h = C.$$

Proposition 10.2. Cooperative generation occurs precisely when two proper principal subcomponents join to the full component.

This formulation places the search inside the finite lattice

$$\mathrm{Sub}_u(C).$$

The absence of cooperative pairs means:

$$C_g \vee C_h = C \Longrightarrow C_g = C \text{ or } C_h = C$$

for every pair $g, h \in C$.

Thus the top element $C$is not the join of two proper singleton-generated subcomponents in the three tested lattices.

This is a lattice-theoretic property of the pointed binary component.

10.4 Closure-Rank Formulation

Define the singleton closure rank

$$r(g) = | \, C_2(\{g\}; u) \, |$$

and the pair closure rank

$$r(g, h) = | \, C_2(\{g, h\}; u) \, |.$$

A singleton generator satisfies

$$r(g) = | \, C \, |.$$

A cooperative pair satisfies

$$r(g) < | \, C \, |,$$

$$r(h) <| C |,$$

but

$$r(g,h) =| C |.$$

Define the pair synergy in cardinality by

$$\mathrm{Syn}_r(g,h) = r(g,h) - \max\{r(g), r(h)\}.$$

For an inherited pair,

$$\mathrm{Syn}_r(g,h) = 0$$

because at least one singleton already has rank $| C |$.

For a cooperative pair,

$$\mathrm{Syn}_r(g,h) > 0.$$

This quantity measures the number of binary functions reached jointly beyond the larger singleton-generated subcomponent.

It does not measure whether the joint gain is redundant with the union of the two singleton closures. A sharper quantity is obtained from set inclusion.

Define

$$\mathrm{Syn}_{\cup}(g,h) =| C_2(\{g,h\}; u) \setminus \left(C_g \cup C_h\right) |.$$

A positive value records functions arising only through compositions that use both primitive ecologies.

Even when

$$r(g,h) <| C |,$$

the pair may exhibit local cooperation by generating functions outside both singleton closures.

The current exhaustive result concerns complete cooperation:

$$r(g,h) = \mid C \mid.$$

It does not assert that all non-generator pairs lack local synergistic expansion.

10.5 Macro Enrichment

Suppose $g \in G(C,u)$and $h \in C \setminus \{g\}$. Then

$$C_2(\{g,h\};u) = C.$$

The pair is preserving automatically.

Its mathematical value lies entirely in changing the exact cost profile.

Define

$$\Delta_T(g;h) = T(\{g\}) - T(\{g,h\}).$$

If

$$\Delta_T(g;h) > 0,$$

then $h$is an effective macro relative to g.

The pair may substantially lower:

$$T,$$
$$S_{\max},$$

or

$$H_{\max},$$

even though its content gain is zero:

$$\mid C_2(\{g,h\};u) \mid - \mid C_2(\{g\};u) \mid = 0.$$

This is conservative macro enrichment.

The best pairs in all three principal landscapes belong to this class.

The observed mechanism is therefore:

complete singleton + high-utility derived operation ⟶ compressed presentation.

It is not:

two incomplete primitives ⟶ new completeness.

10.6 Exhaustive Null Result

The complete pair census tested:

$$120$$

Boolean pairs,

$$276$$

Weak Kleene pairs, and

$$3321$$

Strong Kleene pairs.

Among these,

$$92{,}156{,}620$$

pairs, respectively, preserved the full component.

Every preserving pair contained at least one singleton generator.

Therefore

$$| \mathcal{K}_2(C_B, u_B) | = 0,$$
$$| \mathcal{K}_2(C_W, u_W) | = 0,$$

and

$$| \mathcal{K}_2(C_S, u_S) | = 0.$$

Theorem 10.1. Cooperative-Pair Null Theorem.
For the Boolean meet-negation, Weak Kleene, and Strong Kleene pointed binary term-function components,

$$\mathcal{K}_2(C, u) = \emptyset.$$

Equivalently, every preserving pair contains at least one member of the full singleton-generator set:

$$\{g, h\} \in \mathcal{R}_2(C, u) \Longrightarrow g \in G(C, u) \text{ or } h \in G(C, u).$$

The theorem is exhaustive relative to:

the fixed carrier,
the fixed unary involution,
the fixed binary component,

and

unordered binary signatures of width two.

It is not a statement about all finite algebras or all arities.

10.7 Consequence for Minimal Generators

Because the cooperative-pair sets are empty,

$$\mathrm{MinGen}_{\leq 2}(C, u) = \mathcal{R}_1(C, u)$$

for each principal component.

Since the full singleton-generator census found eight generators in every family,

$$| \mathrm{MinGen}_{\leq 2}(C_B, u_B) | = 8,$$
$$| \mathrm{MinGen}_{\leq 2}(C_W, u_W) | = 8,$$

and

$$| \mathrm{MinGen}_{\leq 2}(C_S, u_S) | = 8.$$

Corollary 10.1. Up to width two, the only inclusion-minimal generating signatures in the three principal components are the eight singleton generators.

Every preserving pair is nonminimal.

This gives a complete bounded minimal-generator classification.

It does not exclude minimal signatures of width three or greater.

10.8 Pair Count Predicted by Singleton Incidence

Let

$$N = | C |, g = | G(C, u) |.$$

If every preserving pair contains a singleton generator, then the preserving pair set consists of all unordered pairs having at least one member in $G(C, u)$, except possibly pairs that fail for some other reason. But any pair containing a singleton generator necessarily preserves C. Hence the number of preserving pairs must be

$$\binom{N}{2} - \binom{N-g}{2}.$$

Equivalently,

$$| \mathcal{R}_2(C, u) | = g(N-g) + \binom{g}{2}.$$

For the principal components,

$$g = 8.$$

Boolean:

$$8(16-8) + \binom{8}{2} = 64 + 28 = 92.$$

Weak Kleene:

$$8(24-8) + 28 = 128 + 28 = 156.$$

Strong Kleene:

$$8(82-8)+28=592+28=620.$$

These values exactly reproduce the exhaustive preserving-pair counts.

Theorem 10.2. Singleton-Incidence Enumeration.
In each principal component,

$$\mathcal{R}_2(C,u)=\{\{g,h\}\subseteq C: g\in G(C,u) \text{ or } h\in G(C,u)\}.$$

Consequently,

$$|\,\mathcal{R}_2(C,u)\,|=\binom{|\,C\,|}{2}-\binom{|\,C\,|-8}{2}.$$

The exhaustive pair census therefore collapses to a simple exact characterization after the cooperative null result is known.

This characterization was discovered computationally and then verified combinatorially.

10.9 Preserving-Pair Density Formula

With $N=|\,C\,|$and $g=|\,G(C,u)\,|$, the preserving-pair density under the cooperative null condition is

$$p_2(C,u)=1-\frac{\binom{N-g}{2}}{\binom{N}{2}}.$$

For fixed $g$and increasing N,

$$p_2(C,u)\sim\frac{2g}{N}.$$

Thus preserving pairs become sparse when the component grows while the singleton-generator count remains bounded.

For $g=8$,

$$p_2=1-\frac{(N-8)(N-9)}{N(N-1)}.$$

Substitution gives the observed densities:

$$p_2(C_B, u_B) = \frac{92}{120},$$
$$p_2(C_W, u_W) = \frac{156}{276},$$

and

$$p_2(C_S, u_S) = \frac{620}{3321}.$$

The decreasing densities are therefore explained exactly by constant singleton-generator count and increasing component size.

No additional pair-generating mechanism compensates for the growing nongenerator region.

10.10 Structure of the Width-Two Graph

Under the cooperative null theorem, every pair node is adjacent to at least one singleton-generator node.

A pair containing two singleton generators has two demotion edges leading to preserving singleton states.

A pair containing exactly one singleton generator has only one preserving demotion.

Let

$$g = \mid G(C,u) \mid, n = \mid C \mid.$$

The number of generator-generator pair nodes is

$$\binom{g}{2}.$$

The number of generator-nongenerator pair nodes is

$$g(n-g).$$

Thus:

$$\mid \mathcal{R}_2 \mid = \binom{g}{2} + g(n-g).$$

For $g = 8$, the first class always contains

$$\binom{8}{2} = 28$$

states.

The second contains:

$$64$$

Boolean states,

$$128$$

Weak Kleene states, and

$$592$$

Strong Kleene states.

The directed edge count follows immediately.

Each generator-generator pair contributes four directed edges:

$$\text{two promotions} + \text{two demotions.}$$

Each generator-nongenerator pair contributes two directed edges:

$$\text{one promotion} + \text{one demotion.}$$

Therefore

$$| E | = 4\binom{g}{2} + 2g(n - g).$$

For Boolean,

$$4(28) + 2(64) = 240.$$

For Weak Kleene,

$$112 + 256 = 368.$$

For Strong Kleene,

$$112 + 1184 = 1296.$$

This exactly explains the graph-edge counts reported in Section 9.

Proposition 10.3. Under the cooperative null condition, the entire width-two preserving graph is determined combinatorially by C, $G(C, u)$, and the exact edge weights.

Its unweighted incidence structure requires no further closure calculation once the singleton-generator set is known.

The exhaustive pair computation remains necessary to establish the absence of cooperative exceptions.

10.11 Connectedness Revisited

Every preserving pair contains a singleton generator. Hence every pair node is connected directly to at least one singleton node.

The singleton-generator nodes are mutually connected through generator-generator pairs:

$$\{g\} \longrightarrow \{g, h\} \longrightarrow \{h\}.$$

Therefore the width-two graph is connected whenever

$$| G(C, u) | \geq 2.$$

Proposition 10.4. If every preserving pair contains a singleton generator and the component has at least two singleton generators, then the preserving width-two graph is connected.

Proof. Every pair is adjacent to a generator singleton. Any two singleton generators $g, h$are connected through the preserving pair $\{g, h\}$. Hence every state lies in one connected component. □

The connectedness observed in Cycle 11 is therefore a consequence of the stronger cooperative null theorem together with the existence of eight singleton generators.

10.12 Negative Results as Discovery Results

A target-free search should not be evaluated only by positive structures it reports.

The cooperative search had a precise positive target:

$$\exists g, h \in C$$

such that

$$C_g < C, C_h < C, C_g \vee C_h = C.$$

The exhaustive computation returned:

$$\neg\exists g, h$$

with those properties in each principal component.

This is not a failure to obtain output. It is a complete finite theorem about the tested presentation spaces.

A disciplined automated discovery system must be capable of returning:

the sought phenomenon is absent here.

Otherwise it is biased toward positive narratives and may reinterpret ordinary macro promotion as emergent cooperation.

The zero result prevents precisely that exaggeration.

10.13 Why Cooperation Would Have Been Stronger

Macro enrichment begins with an already complete generator.

The extensional component is fully present before the second primitive is added:

$$C_2(\{g\}; u) = C.$$

The added operation changes only the metric.

Cooperative generation would instead make completeness depend on interaction:

$$C_2(\{g\}; u) < C,$$
$$C_2(\{h\}; u) < C,$$

but

$$C_2(\{g,h\};u) = C.$$

This would represent a genuine change in minimal generative ontology.

Neither primitive alone would define the full algebraic language. The complete component would be born only at the pair level.

In the context of Algebraic Abiogenesis, this would be closer to a higher-order formation event:

$$\text{partial operational worlds} + \text{interaction} \longrightarrow \text{complete theory component.}$$

The current results do not show this.

They show exact reorganization after completeness already exists.

10.14 Cooperation Defect

The absence of complete cooperation may be quantified.

For nongenerators $g, h \notin G(C,u)$, define the completeness defect

$$D_C(g,h) = |\ C\ | - |\ C_2(\{g,h\};u)\ |.$$

A cooperative pair has

$$D_C(g,h) = 0.$$

The minimum nongenerator-pair defect is

$$D_{\min}(C,u) = \min_{\substack{g,h \notin G(C,u) \\ g \neq h}} D_C(g,h).$$

The cooperative null theorem implies

$$D_{\min}(C,u) > 0.$$

This quantity measures how close the strongest pair of individually incomplete primitives comes to generating the whole component.

The current release establishes positivity but does not center the scientific claims on the exact defect values.

A future analysis should record:

$$D_{\min},$$

the pairs attaining it, and the identities of the missing functions.

Near-cooperative pairs may reveal which operational distinctions prevent full joint generation.

10.15 Cooperative Depth

The width at which cooperation first occurs may be greater than two.

Define the minimal generating width

$$\mu(C,u) = \min\{|\ S\ |: S \subseteq C,\ C_2(S;u) = C\}.$$

Because singleton generators exist,

$$\mu(C,u) = 1$$

for all three principal components.

This scalar does not detect additional minimal generators at larger widths.

Define instead the cooperative width spectrum

$$W_{\min}(C,u) = \{|\ S\ |: S \in \mathrm{MinGen}(C,u)\}.$$

The current results establish

$$W_{\min}(C,u) \cap \{1,2\} = \{1\}.$$

They do not determine whether

$$3 \in W_{\min}(C,u)$$

or whether higher-width minimal generating signatures exist.

A width-three cooperative signature would satisfy

$$C_2(\{g,h,k\};u) = C$$

while every proper subset fails.

Such a signature could coexist with singleton generators elsewhere in the same component.

The absence of cooperative pairs therefore does not imply that all minimal generators are singletons.

10.16 Cooperative Search Beyond Existing Components

The tested Boolean and Kleene systems may be structurally unfavorable places to seek cooperative generation because their full singleton-generator orbits are already large enough to dominate preserving pair incidence.

A stronger search should examine components selected for:

$$| G(C,u) |= 0$$

or very low singleton-generator density.

If

$$G(C,u) = \emptyset,$$

then every preserving pair is automatically cooperative.

The target condition becomes:

$$\mathcal{R}_1(C,u) = \emptyset, \mathcal{R}_2(C,u) \neq \emptyset.$$

This is the cleanest finite signature of irreducible width-two generation.

Definition 10.2. A strictly pair-born component is a pointed component satisfying

$$\mathcal{R}_1(C,u) = \emptyset$$

and

$$\mathcal{R}_2(C,u) \neq \emptyset.$$

Such a component cannot be generated by any single binary primitive relative to u, but can be generated by at least one pair.

A strictly pair-born example would materially extend the present results.

10.17 Search Policy for Pair-Born Components

A target-free pair-cooperation search may proceed as follows.

First, generate or enumerate finite pointed components (C, u).

Second, reject components with:

$$| C |$$

below a declared nontriviality threshold or with degenerate operational behavior.

Third, compute the full singleton-generator set:

$$G(C, u).$$

Fourth, prioritize components with:

$$| G(C, u) | = 0$$

or low generator density.

Fifth, exhaustively test unordered pairs.

Sixth, retain pairs satisfying:

$$C_2(\{g, h\}; u) = C,$$
$$C_2(\{g\}; u) \neq C,$$

and

$$C_2(\{h\}; u) \neq C.$$

Seventh, compute exact size-height profiles and structural certificates for the pair-born component.

The discovery target is not a known circuit identity or named algebra. It is a generative pattern:

$$\text{minimal basis width} = 2.$$

This remains operation-first and target-free at the level of semantic interpretation.

10.18 Cooperation Versus Complementarity

Two primitives may be complementary without jointly generating the full component.

Let

$$C_g = C_2(\{g\}; u), C_h = C_2(\{h\}; u).$$

Their overlap is

$$C_g \cap C_h.$$

Their separate coverage is

$$C_g \cup C_h.$$

Their joint closure is

$$C_{g,h} = C_2(\{g, h\}; u).$$

Define the complementarity gain

$$\Gamma(g, h) = | C_{g,h} | - | C_g \cup C_h |.$$

If

$$\Gamma(g, h) > 0,$$

then interaction between the two primitives creates binary functions unavailable in either singleton ecology, even if

$$C_{g,h} \neq C.$$

Thus complete cooperation is the strongest case:

$$\Gamma(g, h) > 0$$

and

$$C_{g,h} = C.$$

The current null result concerns only the second conjunction.

Partial complementarity may still be abundant among nongenerator pairs.

This suggests a graded research program:

overlap ⟶ partial synergy ⟶ near completeness ⟶ cooperative completeness.

10.19 Cooperation and Primitive Cost

If a cooperative pair exists, neither primitive can be interpreted merely as an optional macro added to an already complete singleton basis.

Its width penalty has a different status.

For an inherited pair, the comparison

$$J_\lambda(\{g,h\})$$

versus

$$J_\lambda(\{g\})$$

asks whether the macro $h$is worth its acquisition cost.

For a strictly cooperative pair, removing either primitive destroys completeness. The relevant comparison is not between two preserving presentations of widths one and two, because no preserving singleton exists.

The width cost becomes unavoidable:

$$\min_{S\in\mathcal{R}(C,u)} |\ S\ | = 2.$$

Thus cooperative generation would separate:

necessary vocabulary width

from

optional vocabulary enrichment.

The present systems exhibit only the latter.

10.20 Scientific Interpretation

The exact pair landscapes establish that a born finite theory may contain many efficient two-primitive presentations.

However, every one of those presentations rests on an already complete singleton generator.

The second primitive can be interpreted as:

a macro,
a promoted derived operation,
a cost-reducing conceptual abbreviation,

or

a new access coordinate.

It cannot be interpreted as one half of an irreducible joint birth.

This distinction sharpens the title and claims of the work.

The release demonstrates exact signature refactoring.

It does not yet demonstrate exact signature abiogenesis at width two.

10.21 Cooperative Null Corollary for the Best Pairs

Let

$$S_2^*$$

be a globally best pair under total minimum size.

Because

$$S_2^* \in \mathcal{R}_2(C, u)$$

and

$$\mathcal{K}_2(C, u) = \emptyset,$$

there exists

$$g^* \in S_2^*$$

such that

$$C_2(\{g^*\}; u) = C.$$

The other member

$$m^* \in S_2^* \setminus \{g^*\}$$

is derivable from g^*.

Hence

$$S_2^* = \{g^*, m^*\}$$

is a complete primitive plus an optimal macro.

Corollary 10.2. The Boolean, Weak Kleene, and Strong Kleene pair optima with totals

$$22{,}36{,}190$$

are macro-enriched singleton bases.

Their improved costs do not result from irreducible cooperation.

This is the correct interpretation of the width-two optima.

10.22 Theorem of Bounded Generative Nonemergence

Theorem 10.3. Within the complete width-two signature spaces of the three principal components, no new minimal generative organization emerges beyond the singleton-generator orbit.

More precisely:

1. every preserving pair contains a singleton generator;
2. every preserving pair is a conservative enrichment of a complete singleton presentation;
3. no pair of proper singleton-generated subcomponents joins to the full component;
4. the bounded minimal-generator family consists only of the eight singleton generators;

5. all pair improvements are representation improvements rather than content-formation improvements.

The theorem is a bounded nonemergence result.

It does not deny that cooperative generation exists in other finite components, higher widths, or higher arities.

It states that the tested systems do not exhibit it.

10.23 Research Consequence

The next major advance should not be another calculation showing that a derived macro lowers expression cost in a component already generated by one primitive.

That phenomenon is now established and classified.

A stronger Stage 3 target is one of the following:

$$\mathcal{R}_1(C,u) = \emptyset, \mathcal{R}_2(C,u) \neq \emptyset,$$

or

$$\mathrm{MinGen}_k(C,u) \neq \emptyset$$

for some

$$k \geq 2$$

with no lower-width preserving subset.

An alternative target is a theorem characterizing when cooperative pairs cannot exist, using the lattice of singleton-generated subcomponents.

Either result would move beyond macro enrichment.

The present null result supplies the baseline against which such a discovery can be recognized.

It prevents ordinary signature widening from being mistaken for emergent generative cooperation and leaves a precise open condition:

$$\exists g, h\big[C_g < C \wedge C_h < C \wedge C_g \vee C_h = C\big].$$

No such pair occurs in the current exact landscapes.

11. Penalized Vocabulary and Optimal Presentation

11.1 The Free-Primitive Problem

Primitive promotion lowers representation cost partly because an added operation is assigned unit cost by definition.

If every function in a finite component were declared primitive, then every nonprojection function would satisfy

$$s_C(f) \leq 1.$$

Consequently,

$$T(C)$$

would be close to the smallest value permitted by the cost convention. Such a result would not reveal a mathematically economical presentation. It would merely transfer complexity from derivations into the primitive vocabulary.

Unpenalized minimization over all preserving signatures is therefore degenerate:

$$\min_{S \in \mathcal{R}(C,u)} T(S)$$

favors increasingly wide signatures.

A meaningful optimization must account for both:

derivational economy

and

primitive economy.

The first rewards short representations.

The second penalizes the acquisition, storage, interpretation, or maintenance of additional primitive operations.

Let

$$w(S) = | S |$$

be the binary primitive width of S.

The simplest penalized objective is

$$J_\lambda(S) = T(S) + \lambda(w(S) - 1),$$

where

$$\lambda \geq 0.$$

The first binary primitive is treated as the baseline. Every additional primitive costs λ.

For singleton signatures,

$$J_\lambda(S) = T(S).$$

For pair signatures,

$$J_\lambda(S) = T(S) + \lambda.$$

The parameter λconverts primitive width and term size into one scalar objective.

11.2 Interpretation of the Primitive Price

The value λmay represent several different costs.

In symbolic computation, it may model:

the implementation cost of an additional operator.

In mathematical exposition, it may model:

the cognitive cost of introducing another primitive concept.

In circuit synthesis, it may model:

the fabrication or library cost of another gate type.

In an automated theory-formation system, it may model:

the description length required to store and certify another operation table.

In a learned representational system, it may model:

the cost of allocating another reusable internal symbol.

The mathematical optimization is independent of which interpretation is chosen. Only the numerical value of λchanges.

Definition 11.1. The λ-optimal preserving signature is any

$$S_\lambda^* \in \operatorname*{arg\,min}_{S \in \mathcal{R}_{\le k}(C,u)} J_\lambda(S).$$

For the present exact landscapes,

$$k = 2.$$

The optimal signature may change as λchanges.

11.3 Width-Constrained Optima

Before introducing a penalty, define the best total cost at each width:

$$\tau_j(C,u) = \min_{S \in \mathcal{R}_j(C,u)} T(S).$$

For the principal components,

Component	τ_1	τ_2
Boolean	30	22
Weak Kleene	48	36
Strong Kleene	291	190

The pair advantage at zero width price is

$$\Delta_{2:1} = \tau_1 - \tau_2.$$

Thus

$$\Delta_{2:1}^B = 8,$$
$$\Delta_{2:1}^W = 12,$$

and

$$\Delta_{2:1}^{S} = 101.$$

These are the maximum total-size savings available by moving from the best singleton presentation to the best pair presentation.

Proposition 11.1. At zero primitive price,

$$\lambda = 0,$$

the globally best pair strictly improves the best singleton in every principal component.

Proof. The exact differences are

$$30 - 22 = 8 > 0,$$
$$48 - 36 = 12 > 0,$$

and

$$291 - 190 = 101 > 0.$$

□

This does not imply that pair signatures remain preferable at positive primitive cost.

11.4 Break-Even Threshold

For a best singleton $S_1^*$and best pair S_2^*,

$$J_\lambda(S_1^*) = \tau_1,$$

whereas

$$J_\lambda(S_2^*) = \tau_2 + \lambda.$$

The pair is preferred precisely when

$$\tau_2 + \lambda < \tau_1.$$

Therefore

$$\lambda < \tau_1 - \tau_2.$$

Definition 11.2. The singleton-pair break-even price is

$$\lambda^* = \tau_1 - \tau_2.$$

At

$$\lambda = \lambda^*,$$

the best singleton and best pair tie.

For

$$\lambda > \lambda^*,$$

the best singleton is preferred.

For

$$0 \leq \lambda < \lambda^*,$$

the best pair is preferred.

The exact thresholds are

$$\lambda_B^* = 8,$$
$$\lambda_W^* = 12,$$

and

$$\lambda_S^* = 101.$$

Theorem 11.1. Exact Width-Penalty Thresholds.
Within the complete preserving-signature spaces of width at most two:

$$S_\lambda^* \in \begin{cases} \mathcal{R}_2(C_B, u_B), & 0 \leq \lambda < 8, \\ \mathcal{R}_1(C_B, u_B) \cup \mathcal{R}_2(C_B, u_B), & \lambda = 8, \\ \mathcal{R}_1(C_B, u_B), & \lambda > 8, \end{cases}$$

$$S_\lambda^* \in \begin{cases} \mathcal{R}_2(C_W, u_W), & 0 \le \lambda < 12, \\ \mathcal{R}_1(C_W, u_W) \cup \mathcal{R}_2(C_W, u_W), & \lambda = 12, \\ \mathcal{R}_1(C_W, u_W), & \lambda > 12, \end{cases}$$

and

$$S_\lambda^* \in \begin{cases} \mathcal{R}_2(C_S, u_S), & 0 \le \lambda < 101, \\ \mathcal{R}_1(C_S, u_S) \cup \mathcal{R}_2(C_S, u_S), & \lambda = 101, \\ \mathcal{R}_1(C_S, u_S), & \lambda > 101. \end{cases}$$

The result is exact relative to the frozen binary components and width bound.

11.5 Piecewise-Linear Lower Envelope

Every preserving signature defines an affine cost function of λ:

$$L_S(\lambda) = T(S) + \lambda(|\, S \,| - 1).$$

For a singleton,

$$L_S(\lambda) = T(S),$$

a horizontal line.

For a pair,

$$L_S(\lambda) = T(S) + \lambda,$$

a line of slope one.

The optimum is the lower envelope

$$J^*(\lambda) = \min_{S \in \mathcal{R}_{\le 2}(C,u)} L_S(\lambda).$$

Because all singleton lines have the same slope zero, only the singleton with minimum Tcan lie on the lower envelope.

Because all pair lines have the same slope one, only pairs with minimum Tcan lie on the lower envelope.

Therefore the entire width-two penalized optimization collapses to comparison of two numbers:

$$\tau_1 \text{and} \tau_2 + \lambda.$$

Proposition 11.2. Under a uniform primitive price and a width bound of two,

$$J^*(\lambda) = \min\{\tau_1, \tau_2 + \lambda\}.$$

Thus the lower envelope contains at most one transition point.

For the principal components,

$$J_B^*(\lambda) = \min\{30{,}22 + \lambda\},$$
$$J_W^*(\lambda) = \min\{48{,}36 + \lambda\},$$

and

$$J_S^*(\lambda) = \min\{291{,}190 + \lambda\}.$$

Equivalently,

$$J_B^*(\lambda) = \begin{cases} 22 + \lambda, & 0 \le \lambda \le 8, \\ 30, & \lambda \ge 8, \end{cases}$$
$$J_W^*(\lambda) = \begin{cases} 36 + \lambda, & 0 \le \lambda \le 12, \\ 48, & \lambda \ge 12, \end{cases}$$

and

$$J_S^*(\lambda) = \begin{cases} 190 + \lambda, & 0 \le \lambda \le 101, \\ 291, & \lambda \ge 101. \end{cases}$$

At the transition value, both branches are optimal.

11.6 Multiplicity of Optimal Signatures

The aggregate optimum need not be realized by a unique signature.

Let

$$M_j = \left| \arg\min_{S \in \mathcal{R}_j(C,u)} T(S) \right|$$

be the multiplicity of width-j total-cost optima.

The Stage 2 landscapes contain several symmetry-related optimal signatures.

For the best singleton profile types, the multiplicity is four in each principal component.

For the globally best pair total:

$$M_2^B = 8,$$
$$M_2^W = 4,$$

and

$$M_2^S = 8.$$

Therefore, below the break-even threshold, the complete optimal set consists of the best-pair orbit or profile class. Above the threshold, it consists of the best-singleton class.

At the threshold,

$$\lambda = \lambda^*,$$

both optimal classes coexist.

Definition 11.3. The optimal-signature correspondence is

$$\mathcal{O}(\lambda) = \underset{S \in \mathcal{R}_{\leq 2}(C,u)}{\arg\min}\ J_\lambda(S).$$

Unlike a single-valued optimizer, $\mathcal{O}(\lambda)$retains all symmetry-related optima.

11.7 Primitive Price and Macro Utility

Let $g$be a singleton generator and $m$an added macro.

The pair

$$\{g, m\}$$

is preferred to $\{g\}$when

$$T(\{g, m\}) + \lambda < T(\{g\}).$$

Equivalently,

$$\Delta_T(g; m) > \lambda,$$

where

$$\Delta_T(g; m) = T(\{g\}) - T(\{g, m\}).$$

Thus each macro has a local break-even price

$$\lambda^*(g; m) = \Delta_T(g; m).$$

The global pair threshold is the maximum achievable local gain after allowing the singleton basis itself to vary:

$$\lambda^* = \tau_1 - \tau_2.$$

The difference between local and global macro value is important.

A macro may be highly useful relative to the original primitive but less useful relative to a better singleton basis.

Conversely, the globally best pair may require first replacing the original primitive.

Therefore

$$\lambda^*$$

is a property of the entire width-two landscape, not merely of one promotion from the initial signature.

11.8 Original-Anchored Thresholds

Let

$$S_0 = \{b_0\}$$

be the original primitive, and let

$$S_{2,0}^*$$

be the best pair containing b_0.

The anchored break-even value is

$$\lambda_0^* = T(S_0) - T(S_{2,0}^*).$$

The exact values are

$$\lambda_{0,B}^* = 41 - 25 = 16,$$
$$\lambda_{0,W}^* = 69 - 44 = 25,$$

and

$$\lambda_{0,S}^* = 444 - 223 = 221.$$

These thresholds are much larger than the best-singleton-to-best-pair thresholds:

$$16 > 8,$$
$$25 > 12,$$
$$221 > 101.$$

This does not mean the anchored pair is globally better. It means the original singleton is comparatively inefficient, so adding one macro to it produces a large apparent gain.

The correct width comparison depends on the baseline.

Proposition 11.3. Macro value measured relative to an inefficient original basis may overstate the value of width expansion relative to the best available singleton basis.

The exact decomposition is

$$T(S_0) - T(S_2^*) = [T(S_0) - T(S_1^*)] + [T(S_1^*) - T(S_2^*)].$$

For Boolean,

$$41 - 22 = (41 - 30) + (30 - 22) = 11 + 8.$$

For Weak Kleene,

$$69 - 36 = 21 + 12.$$

For Strong Kleene,

$$444 - 190 = 153 + 101.$$

The first term is basis-replacement gain.

The second is genuine best-singleton-to-best-pair macro gain.

This separates primitive correction from vocabulary widening.

11.9 Refactoring Cost Versus Steady-State Cost

The objective

$$J_\lambda(S)$$

describes steady-state presentation cost. It does not include the one-time cost of moving from the current signature to S.

Let S_0be the current presentation and let

$$d_{\mathrm{acq}}(S_0, S)$$

be the minimum cumulative promotion-definition cost of a preserving path from $S_0$to S.

Define the transition-aware objective

$$K_{\lambda,\gamma}(S \mid S_0) = J_\lambda(S) + \gamma\, d_{\mathrm{acq}}(S_0, S),$$

where

$$\gamma \geq 0$$

weights one-time refactoring cost relative to steady-state use.

If the presentation will be used only briefly, a low-access-cost endpoint may not repay its transition cost.

If the presentation will be reused many times, steady-state savings dominate.

Let Rbe the expected number of repeated uses of the full component. Define

$$K_{\lambda,R}(S \mid S_0) = d_{\mathrm{acq}}(S_0, S) + R\, J_\lambda(S).$$

A refactoring from $S_0$to $S$is favorable when

$$d_{\mathrm{acq}}(S_0, S) + R\,J_\lambda(S) < R\,J_\lambda(S_0).$$

Hence the minimum reuse count required to justify the transition is

$$R > \frac{d_{\mathrm{acq}}(S_0, S)}{J_\lambda(S_0) - J_\lambda(S)}$$

whenever the denominator is positive.

Definition 11.4. The amortization threshold of Srelative to $S_0$is

$$R^*(S_0 \to S; \lambda) = \frac{d_{\mathrm{acq}}(S_0, S)}{J_\lambda(S_0) - J_\lambda(S)}.$$

This adds a temporal dimension to the landscape.

A mathematically optimal language may be suboptimal for one-time use but optimal under repeated reuse.

11.10 Amortization of the Global Pair Refactorings

At zero primitive price, the best pair improvements from the original signatures are:

$$19{,}33{,}254.$$

The minimum cumulative promotion-definition costs of the shortest preserving paths are:

$$6{,}4{,}8.$$

Therefore the zero-price amortization thresholds are

$$R_B^* = \frac{6}{19} \approx 0.316,$$
$$R_W^* = \frac{4}{33} \approx 0.121,$$

and

$$R_S^* = \frac{8}{254} = \frac{4}{127} \approx 0.0315.$$

Since one complete use corresponds to

$$R \geq 1,$$

all three global refactorings repay their promotion-definition cost within one full aggregate use of the component under the declared metric.

This conclusion depends on treating the recorded promotion-definition size as the complete transition cost. Human interpretation, implementation, proof, or communication costs may be much larger.

The finite statement is only:

$$d_{\mathrm{acq}} < T(S_0) - T(S_2^*)$$

in all three examples.

11.11 Height-Penalized Optimization

Total node count is not the only objective.

Define

$$J_{\lambda,\mu,\nu}(S) = T(S) + \lambda(|\ S\ | - 1) + \mu S_{\max}(S) + \nu H_{\max}(S),$$

with

$$\lambda, \mu, \nu \geq 0.$$

The pairwise difference between signatures $S$and $T$is

$$\begin{aligned} &J_{\lambda,\mu,\nu}(S) - J_{\lambda,\mu,\nu}(T) \\ &= T(S) - T(T) + \lambda(|\ S\ | - |\ T\ |) + \mu(S_{\max}(S) - S_{\max}(T)) + \nu(H_{\max}(S) - H_{\max}(T)). \end{aligned}$$

Each pair of candidate signatures defines a hyperplane in parameter space where they tie.

The parameter space is partitioned into polyhedral regions, each associated with one or more optimal signatures.

For width at most two and three scalar cost criteria, the exact refactoring landscape induces a finite phase diagram.

Definition 11.5. The presentation phase region of $S$is

$$\mathcal{P}(S) = \left\{(\lambda, \mu, \nu) \in \mathbb{R}^3_{\geq 0} : S \in \arg\min_T J_{\lambda,\mu,\nu}(T)\right\}.$$

The scalar break-even thresholds of Section 11.4 are one-dimensional slices:

$$\mu = \nu = 0.$$

11.12 Strong Kleene Total-Size Versus Radius Tradeoff

The Strong Kleene pair frontier contains at least two relevant aggregate pair types:

$$P_T = (2{,}190{,}5{,}3)$$

and

$$P_R = (2{,}193{,}4{,}3),$$

where the coordinates are

$$(|\ S\ |, T, S_{\max}, H_{\max}).$$

Both signatures have the same width and saturation height.

Their objective difference under $J_{\lambda,\mu,\nu}$is

$$J(P_R) - J(P_T) = 193 - 190 + \mu(4 - 5) + \nu(3 - 3).$$

Hence

$$J(P_R) - J(P_T) = 3 - \mu.$$

Therefore:

$$\mu < 3 \Longrightarrow P_T \text{ is preferred,}$$
$$\mu > 3 \Longrightarrow P_R \text{ is preferred,}$$

and

$$\mu = 3$$

produces a tie.

Proposition 11.4. In the Strong Kleene landscape, one unit of saturation-size reduction costs exactly three units of total minimum size at the pair frontier.

This is an exact exchange rate under the declared aggregate metric.

The width price λcancels because both candidates have width two. The height price νcancels because both have saturation height three.

11.13 Boolean and Weak Kleene Height Tradeoffs

For the Boolean singleton frontier, compare:

$$B_T = (1,30,5,4)$$

and

$$B_H = (1,37,5,3).$$

Their difference under the height-penalized objective

$$T + \nu H_{\max}$$

is

$$(37 + 3\nu) - (30 + 4\nu) = 7 - \nu.$$

Therefore the lower-height singleton is preferred when

$$\nu > 7.$$

The two tie at

$$\nu = 7.$$

For Weak Kleene, compare:

$$W_T = (1,48,5,4)$$

and

$$W_H = (1,63,5,3).$$

The difference is

$$(63 + 3\nu) - (48 + 4\nu) = 15 - \nu.$$

Thus the lower-height singleton is preferred when

$$\nu > 15.$$

These thresholds quantify the price required to justify sacrificing total node economy for one unit of worst-case depth reduction.

11.14 Pair Versus Low-Height Singleton

The best pair has lower saturation height than either singleton frontier in Boolean and Weak Kleene:

$$H_{\max} = 2$$

rather than

$$3 \text{or} 4.$$

For Boolean, compare the best pair

$$P_B = (2,22,3,2)$$

with the low-height singleton

$$B_H = (1,37,5,3).$$

Under

$$J_{\lambda,\mu,\nu},$$

the pair is preferred when

$$22 + \lambda + 3\mu + 2\nu < 37 + 5\mu + 3\nu.$$

Equivalently,

$$\lambda < 15 + 2\mu + \nu.$$

Against the minimum-total singleton

$$B_T = (1{,}30{,}5{,}4),$$

the pair is preferred when

$$\lambda < 8 + 2\mu + 2\nu.$$

Thus penalties on saturation size and height enlarge the parameter region in which the pair is optimal.

The corresponding Weak Kleene inequalities are:

against $W_H = (1{,}63{,}5{,}3)$,

$$\lambda < 27 + 2\mu + \nu;$$

against $W_T = (1{,}48{,}5{,}4)$,

$$\lambda < 12 + 2\mu + 2\nu.$$

The scalar thresholds $8$and $12$are therefore conservative slices that ignore the pair's simultaneous radius improvements.

11.15 Primitive Description Length

Uniform width cost treats every additional primitive as equally expensive. Exact operation tables may have different description lengths relative to a chosen coding language.

Let

$$K(m)$$

be the description cost of primitive m.

Define

$$J_K(S) = T(S) + \sum_{m \in S \setminus S_{\text{base}}} K(m).$$

Possible choices include:

$$K(m) = \text{raw table length},$$
$$K(m) = \text{compressed table length},$$
$$K(m) = s_{S_0}(m),$$

or

$$K(m) = \text{formal proof length of its defining equation.}$$

If

$$K(m) = s_{S_0}(m),$$

then a primitive is charged according to the exact cost of defining it in the original language.

This produces a nonuniform acquisition model.

The optimization becomes

$$S^* \in \arg\min_{S \in \mathcal{R}(C,u)} \left[T(S) + \sum_{m \in S} K(m) \right].$$

Uniform $\lambda$is the special case

$$K(m) = \lambda$$

for every added primitive.

11.16 Minimal Description Presentation

The theory itself may be encoded by:

1. the chosen primitive tables;

2. a rule identifying the fixed unary operation;
3. the derivations required to reconstruct the component;
4. the certificate that the signature preserves the component.

A crude two-part description length is

$$L(S) = L_{\mathrm{prim}}(S) + L_{\mathrm{access}}(C \mid S).$$

Set

$$L_{\mathrm{access}}(C \mid S) = T(S).$$

If primitive tables are encoded directly,

$$L_{\mathrm{prim}}(S) = \sum_{b \in S} \ell(\mathrm{tab}(b)).$$

For a fixed carrier and raw coding, every binary operation table has the same length. This reduces to a uniform width penalty.

Under compressed or structure-sensitive coding, primitive costs differ.

The optimal signature then resembles a minimum-description-length presentation:

$$S^* \in \arg\min_S L(S).$$

This interpretation connects exact signature refactoring to mathematical compression without claiming that the chosen code length is uniquely canonical.

11.17 Robust Optimization

A chosen value of λmay be uncertain.

Suppose only

$$\lambda \in [\lambda_-, \lambda_+]$$

is known.

A robust signature minimizes worst-case penalized cost:

$$S^*_{\text{rob}} \in \arg\min_S \max_{\lambda\in[\lambda_-,\lambda_+]} J_\lambda(S).$$

Since $J_\lambda(S)$is nondecreasing in λ,

$$\max_{\lambda\in[\lambda_-,\lambda_+]} J_\lambda(S) = J_{\lambda_+}(S).$$

Thus robust optimization under an interval uncertainty reduces to using the upper price bound.

A minimax-regret objective is different.

Define

$$\text{Reg}(S;\lambda) = J_\lambda(S) - J^*(\lambda).$$

Then choose

$$S^*_{\text{reg}} \in \arg\min_S \max_{\lambda\in[\lambda_-,\lambda_+]} \text{Reg}(S;\lambda).$$

This may favor a presentation that is never uniquely optimal but remains close to optimal across the uncertainty range.

The exact finite landscape makes such secondary optimization computable.

11.18 Sensitivity to the Width Bound

The current objective is optimized over

$$\mathcal{R}_{\le 2}(C,u).$$

Let

$$\tau_k = \min_{S\in\mathcal{R}_k(C,u)} T(S).$$

For arbitrary width bound K,

$$J^*_K(\lambda) = \min_{1\le k\le K} [\tau_k + \lambda(k-1)].$$

This is the lower envelope of Kaffine functions with slopes

$$0,1,\dots,K-1.$$

As λdecreases, wider signatures may become optimal.

As λincreases, optimal width cannot increase.

Proposition 11.5. Let $k^*(\lambda)$be the smallest optimal width. Then $k^*(\lambda)$is nonincreasing in λ.

Proof. Each width-k branch has slope $k-1$. A branch of larger slope cannot become newly optimal as λincreases after being dominated by a branch of smaller slope. □

The width-two case contains only one possible width transition:

$$2 \longrightarrow 1.$$

Higher-width analysis may produce several phase transitions.

11.19 Concavity of the Optimal Value

The function

$$J_K^*(\lambda) = \min_k [\tau_k + \lambda(k-1)]$$

is the pointwise minimum of affine functions.

Therefore it is piecewise linear and concave.

Its slope at a differentiable point is

$$k^*(\lambda) - 1,$$

the number of additional primitives in the unique optimal width.

Thus the derivative of the optimal-value function records optimal vocabulary expansion.

At a transition point, the derivative is discontinuous because two widths tie.

For the principal width-two landscapes, the slopes are:

$$1$$

below the break-even threshold and

$$0$$

above it.

The kink occurs at:

$$8,12,101.$$

These values are phase boundaries in primitive-price space.

11.20 Optimality Certificates

A finite optimization result should include a certificate stronger than the identity of the reported winner.

For a claimed λ-optimal signature S^*, the certificate contains:

$$J_\lambda(S^*),$$

and for every competing preserving signature S,

$$J_\lambda(S) - J_\lambda(S^*) \geq 0.$$

Because the landscape is finite, this inequality can be checked exhaustively.

For interval claims, such as

$$S^* \text{ is optimal for } 0 \leq \lambda < 8,$$

the affine difference

$$L_S(\lambda) - L_{S^*}(\lambda)$$

is checked over the interval.

Since the difference is affine, it suffices to inspect interval endpoints unless the slopes are equal.

Definition 11.6. A penalized-optimality certificate is the set of affine dominance inequalities establishing that no preserving competitor has lower objective over the declared parameter region.

This converts presentation preference into a checkable finite theorem.

11.21 The Exact Optimization Theorem

Theorem 11.2. Exact Penalized-Presentation Theorem.
For the complete width-two preserving-signature landscapes of the Boolean meet-negation, Weak Kleene, and Strong Kleene pointed binary components:

1. the minimum singleton total costs are 30, 48, and 291;
2. the minimum pair total costs are 22, 36, and 190;
3. the exact uniform primitive-price break-even thresholds are 8, 12, and 101;
4. below each threshold, a globally best pair minimizes

$$J_\lambda(S) = T(S) + \lambda(|\, S \,| - 1);$$

5. above each threshold, a globally best singleton minimizes J_λ;
6. at the threshold, the best singleton and best pair tie;
7. the optimal-value functions are

$$\min\{30, 22 + \lambda\},$$
$$\min\{48, 36 + \lambda\},$$

and

$$\min\{291, 190 + \lambda\};$$

8. the original-anchored macro gains decompose into basis-replacement savings plus true width-two savings;
9. multiobjective penalties generate exact phase boundaries among total-cost, saturation-size, and saturation-height optima.

The theorem turns primitive selection into an exact finite variational problem.

11.22 Interpretation

The unpenalized pair optimum answers:

Which two binary primitives make the component cheapest to reconstruct?

The penalized optimum answers:

Is the reduction in derivational cost worth the additional primitive?

The answer depends on λ.

For Boolean algebra, the second primitive is justified when its price is less than eight aggregate operation nodes.

For Weak Kleene, the threshold is twelve.

For Strong Kleene, the threshold is 101.

The large Strong Kleene threshold reflects the broad reuse value of the optimal added primitive across an eighty-two-function component.

This does not imply that the primitive is intrinsically worth 101 units in every interpretation. It means that under the exact aggregate node-count metric, the best pair saves 101 nodes relative to the best singleton reconstruction of the complete binary component.

The central conclusion is therefore conditional and exact:

$$\text{optimal primitive language} = \text{function of the declared cost policy.}$$

There is no presentation optimum independent of what is being charged.

11.23 From Theory Formation to Variational Language

Stage I treated laws and quotients as emergent from finite operational regularities.

The present optimization treats primitive language as selected from a finite ecology of content-preserving alternatives.

The final variational principle is

$$S^* \in \arg\min_{S \in \mathcal{R}(C,u)} [\mathrm{AccessCost}(C \mid S) + \mathrm{VocabularyCost}(S)].$$

Under the current finite implementation,

$$\mathrm{AccessCost}(C \mid S) = \sum_{f \in C} s_S(f),$$

and

$$\mathrm{VocabularyCost}(S) = \lambda(\mid S \mid -1).$$

The resulting sequence is

anonymous operations ⟶ discovered laws ⟶ certified component ⟶ preserving signatures ⟶ exact cost landscape ⟶ policy-optimal language.

A theory is first born extensionally.

Its language is then selected variationally.

12. Reproducibility, Scope, and Open Problems

12.1 Exact Finite Evidence

The present results are finite and computationally exact within declared search spaces.

Every binary operation is represented by its complete function table. Every candidate term is evaluated on every assignment. Every component-preservation decision is determined by equality of finite function sets. Every minimum-size and minimum-height value is obtained by terminating finite search.

The principal computational claims are therefore not statistical estimates.

They include:

$$531{,}441$$

labeled order-three systems examined in the exhaustive census;

$$83{,}638$$

systems surviving the declared admissibility filters;

$$13{,}965$$

eligible isomorphism classes;

$$122$$

binary functions tested in the complete singleton-generator scans;

$$24$$

singleton generators identified across the three principal components;

$$3{,}717$$

unordered pair signatures examined;

$$868$$

component-preserving pairs;

$$892$$

preserving singleton and pair states;

and

$$1{,}904$$

directed promotion or demotion edges.

These counts are exact relative to the frozen carrier sets, unary operations, binary components, term-cost conventions, filters, and signature-width bounds.

The finite evidence does not establish unrestricted mathematical universality. It establishes complete results inside explicitly delimited computational worlds.

12.2 Repository Structure

The computational record is preserved in the public repository

DougYouvan/algebraic-abiogenesis-stage2.

The repository separates:

source code,
tests,
schemas,
documentation,
frozen results,

and

verification metadata.

The source package contains the operation-table, closure, profile, promotion, and landscape algorithms.

The test suite checks finite invariants, regression conditions, package behavior, and selected independent oracle calculations.

The schemas define the expected structure of computational records.

The documentation reports the experimental cycles and their claim boundaries.

The frozen result bundle preserves the larger evidence collection without requiring hundreds of individual browser-uploaded files.

The verification script reconstructs the intended evidence state and checks the recorded manifests.

This organization distinguishes executable method from frozen output.

12.3 Manifest Verification

Let

$$F_1, \ldots, F_m$$

be the frozen evidence files.

For each F_i, define its digest

$$d_i = \mathrm{SHA256}(F_i).$$

A release manifest records the pairs

$$(F_i, d_i).$$

Verification recomputes every digest and checks

$$\mathrm{SHA256}(F_i) = d_i.$$

The release contains seventy verified manifests.

Manifest agreement establishes that the evidence files have not changed relative to the declared release state.

It does not establish that the algorithms or mathematical interpretations are correct. It establishes archival integrity.

This distinction is essential:

$$\text{checksum validity} \neq \text{mathematical validity}.$$

Checksums answer whether the evidence changed.

Independent recomputation answers whether the evidence can be regenerated.

Proof and code review answer whether the finite procedures justify the claims.

12.4 Reproduction and Self-Certification

A reproduction process must not rewrite the frozen evidence and then generate a new manifest from the rewritten state. Such a procedure would certify only internal consistency of the newly generated files.

The corrected release separates:

$$\text{frozen evidence verification}$$

from

$$\text{fresh recomputation}.$$

Frozen verification treats the recorded manifests as immutable reference data.

Fresh recomputation writes into a separate output location or compares normalized results in memory.

The resulting relation is

$$\text{fresh normalized result} = \text{frozen normalized result}.$$

Only after this comparison is successful may reproduction be said to confirm the scientific record.

Definition 12.1. A non-self-certifying reproduction is a computation in which the reference digests and reference scientific outputs remain fixed throughout the verification process.

The release adopts this principle.

It prevents the verification procedure from redefining success after execution.

12.5 Independent Recalculation

Several principal results were independently recomputed during evaluation of the release.

The Boolean width-two census reproduced:

$$120$$

tested pairs,

$$92$$

preserving pairs,

$$0$$

cooperative pairs,

and best pair statistics

$$T = 22, S_{\max}/H_{\max} = 3/2.$$

The Weak Kleene census reproduced:

$$276$$

tested pairs,

$$156$$

preserving pairs,

$$0$$

cooperative pairs,

and

$$T = 36, S_{\max}/H_{\max} = 3/2.$$

The Strong Kleene independent oracle verified the declared winning pair:

$$| C | = 82,$$
$$T = 190,$$
$$S_{\max} = 5,$$

and

$$H_{\max} = 3.$$

The full Strong Kleene pair census is the most computationally expensive component of Cycle 11. Its frozen exhaustive record reports:

$$3{,}321$$

tested pairs and

$$620$$

preserving pairs.

The complete count is preserved in the release evidence, while targeted independent recomputation verifies the declared optimum and profile.

This difference should remain explicit. Independent verification of the winner is not identical to independent re-enumeration of every candidate pair.

12.6 Test Coverage

The repository’s automated test suite contains thirty-seven passing tests under the evaluated release.

The tests cover:

operation-table validity,
closure behavior,
size and height semantics,
carrier relabeling,

manifest integrity,
cycle-level invariants,

and

selected independent recomputations.

The test suite also passes when warnings are treated as errors.

A passing test suite establishes agreement with the encoded specifications.

It does not establish that the specifications exhaust every possible defect.

In particular, expensive exhaustive claims require dedicated reproduction runs in addition to ordinary unit tests.

The distinction is:

unit verification

versus

full scientific recomputation.

Both are required for strong reproducibility.

12.7 Sharded Exhaustive Computation

The Strong Kleene pair census is sufficiently expensive that uninterrupted monolithic execution creates avoidable risk.

A more robust design partitions the pair space into shards.

Let

$$P = \{\{g, h\} \subseteq C : g \neq h\}.$$

Partition

$$P = P_1 \sqcup \cdots \sqcup P_r.$$

Each shard $P_i$independently records:

its pair-code interval,
all preserving pairs,
local cooperative-pair count,
local minimum total cost,
local Pareto states,

and

its cryptographic digest.

A deterministic merger verifies:

$$P_i \cap P_j = \emptyset$$

for

$$i \neq j,$$

and

$$\bigcup_i P_i = P.$$

It then computes the global counts and optima.

Definition 12.2. A shard certificate is a record proving that a declared subset of the candidate space was evaluated completely under the frozen policy.

Sharding changes no mathematics. It improves fault tolerance, restartability, and independent distribution of computation.

12.8 Computational Scope

The principal exact analyses are restricted to finite carriers and binary term-function components.

For a carrier of cardinality q, the full binary operation space has cardinality

$$q^{q^2}.$$

For

$$q = 3,$$

this is

$$3^9 = 19{,}683.$$

The ternary operation space has cardinality

$$q^{q^3}.$$

For

$$q = 3,$$

this becomes

$$3^{27} = 7{,}625{,}597{,}484{,}987.$$

Thus exact extensional enumeration becomes difficult rapidly with increasing arity.

The Stage 2 release avoids claiming tractability beyond the tested finite regimes.

The framework is general in definition:

$$C_n(S;U) = \mathrm{Cl}_n(U,S),$$

but exact exhaustive execution is strongly constrained by

$$q, n, |\ C_n\ |, |\ S\ |,$$

and the number of candidate signatures.

12.9 Binary-Component Limitation

The preservation relation used throughout is

$$C_2(S;u) = C_2(T;u).$$

This does not establish

$$C_n(S;u) = C_n(T;u)$$

for all

$$n \geq 1.$$

It therefore does not establish equality of complete finitary clones.

The correct interpretation is:

binary operational content is preserved.

The stronger statement

$$\mathrm{Clo}(u,S) = \mathrm{Clo}(u,T)$$

would require either:

higher-arity exhaustive computation,
a clone-theoretic proof,

or

a theorem reducing full clone equality to bounded arity

for the particular algebraic family.

No such unrestricted conclusion is drawn here.

12.10 Finite Model Limitation

The discovery process begins with finite operation tables.

A law recurring across finite models may define an infinite variety, but finite recurrence alone does not prove universal validity.

The transition from finite evidence to general mathematics requires one or more of:

formal derivation,
normal-form proof,
representation theorem,
finite basis theorem,

model-theoretic argument,

or

formal proof-assistant verification.

The Stage I ternary calibration achieved such a transition for the law

$$m(x, y, y) = x$$

by deriving a terminating reduction rule and exact model-count formula.

The Stage 2 signature results are different. Their primary objects are finite components and finite weighted presentation landscapes.

Their exactness is already complete at the finite level.

Generalization to infinite families remains a separate theorem problem.

12.11 Historical Novelty Limitation

A structure may be newly discovered by the computational process while already known in mathematical literature.

The recovered Boolean, Weak Kleene, Strong Kleene, and BCK-related systems are not claimed as historically new algebras.

Likewise, minimum term-size, basis selection, clone generation, functional completeness, and macro introduction have substantial prior literatures.

The defensible contribution lies in the particular integrated construction:

operation-first theory formation

followed by

exact complete finite signature landscapes,
joint size-height profiles,
full singleton-generator spectra,
cooperative-pair classification,
shortest preserving refactoring paths,

and

penalized primitive optimization.

Even this integrated contribution should be evaluated against specialist literature.

Computational rediscovery establishes methodological capability.

Historical priority requires independent review.

12.12 Interpretation Limitation

The system can determine that an operation table generates a component with a particular exact profile.

It does not automatically determine the most meaningful human interpretation of the operations.

For example, a recovered table may later be recognized as:

Boolean implication,
Kleene conjunction,
a polarity transform,

or

an operation in a known variety.

The assignment of those names remains interpretive.

The finite computation can support interpretation by providing:

truth tables,
identity profiles,
generator relations,
symmetry orbits,

and

countermodels.

It cannot guarantee that the first semantic description proposed by a language model or human reader is historically or mathematically correct.

Interpretation must remain separable from extensional evidence.

12.13 Policy Dependence

Every discovery and optimization result is relative to declared policies.

These include:

carrier bounds,
operation arities,
term-generation bounds,
degeneracy filters,
explanation-subtraction rules,
structural certificates,
cost functions,
primitive prices,

and

tie-breaking rules.

Changing a policy may change the selected winner.

This does not make the results arbitrary. It makes them conditional.

The appropriate statement is:

under policy Π, signature S^* is optimal.

The policy itself becomes part of the mathematical specification.

Definition 12.3. A discovery policy is a tuple

$$\Pi = \left(\Pi_{\text{space}}, \Pi_{\text{filter}}, \Pi_{\text{identity}}, \Pi_{\text{certificate}}, \Pi_{\text{cost}}, \Pi_{\text{tie}}\right).$$

A reproducible claim records both:

$$\Pi$$

and the resulting finite output.

Policy-free claims are avoided.

12.14 Metric Dependence

The principal representation metric charges:

$$1$$

for every unary or binary operation occurrence and

$$0$$

for variables.

Other metrics are possible.

A weighted metric may assign:

$$w_u$$

to unary application and

$$w_b$$

to binary primitive b.

Then

$$s_w(u(t)) = s_w(t) + w_u,$$

and

$$s_w(b(t_1, t_2)) = s_w(t_1) + s_w(t_2) + w_b.$$

A circuit metric may charge fan-out, wiring, or gate delay.

A symbolic metric may charge repeated subexpression expansion differently from shared directed acyclic graph representation.

A proof metric may count inference steps rather than syntax nodes.

The exact landscape changes with the metric.

The current results establish one canonical simple baseline:

unit operation-node cost

and

unit nesting depth.

They do not establish metric-independent optimality.

12.15 Tree Terms Versus Shared Circuits

The size function counts syntax-tree nodes.

If the same subterm appears twice, both occurrences are charged.

For example,

$$b(t, t)$$

has size

$$2s(t) + 1.$$

A directed acyclic graph representation could compute $t$once and reuse it. Its circuit size may therefore be smaller.

Define:

$$s_{\text{tree}}(f)$$

for minimum term-tree size and

$$s_{\text{dag}}(f)$$

for minimum shared-circuit size.

Then

$$s_{\text{dag}}(f) \leq s_{\text{tree}}(f).$$

The present release computes

$$s_{\text{tree}}.$$

A circuit-aware extension would require state spaces that preserve reusable intermediate functions and count shared subcomputations.

This extension is especially relevant to the long-term goal of operation-first circuit discovery.

12.16 Target-Free Circuit Discovery

A central future objective is to discover useful logic circuits without supplying the known target identities.

The search should begin from:

anonymous finite gates,
composition rules,

and

exact behavioral equivalence.

It should not be given:

the NAND definitions of AND, OR, or NOT,

or a predetermined desired circuit.

A target-free circuit stage may proceed by:

1. generating finite gate compositions;
2. identifying exact functional collisions;
3. clustering recurrent operational components;
4. detecting minimal generating signatures;
5. promoting useful derived gates;
6. optimizing tree and shared-circuit complexity;

7. searching for cooperative gate sets;
8. comparing discovered components with known logical clones only after formation.

A particularly strong result would be a strictly pair-born functional component:

$$\mathcal{R}_1(C,u) = \emptyset,$$
$$\mathcal{R}_2(C,u) \neq \emptyset.$$

This would demonstrate irreducible joint generation rather than macro enrichment.

12.17 Search for Cooperative Components

The cooperative null result identifies a precise next target.

Search for a finite pointed component satisfying

$$G(C,u) = \emptyset$$

and

$$\mathcal{R}_2(C,u) \neq \emptyset.$$

Equivalently, find

$$g, h \in C$$

such that

$$C_2(\{g\};u) < C,$$
$$C_2(\{h\};u) < C,$$

but

$$C_2(\{g,h\};u) = C.$$

A complete result should include:

the finite carrier,
the unary background,
the operation tables,
the singleton closures,

the pair closure,
minimality proof,
exact size-height profile,

and

structural interpretation or novelty audit.

A strictly pair-born component would be the clearest finite advance beyond the current release.

12.18 Higher-Width Minimal Generators

Even when singleton generators exist, there may be inclusion-minimal generating signatures of width three or greater.

Define

$$\mathrm{MinGen}_k(C,u) = \{S \subseteq C : |\ S\ | = k,\ C_2(S;u) = C,\ \forall T \subsetneq S,\ C_2(T;u) \neq C\}.$$

The current results establish:

$$\mathrm{MinGen}_2(C,u) = \emptyset$$

for the three principal components.

They do not determine

$$\mathrm{MinGen}_3(C,u),$$
$$\mathrm{MinGen}_4(C,u),$$

or higher widths.

A full minimal-generator hypergraph would have vertex set C and hyperedges

$$\mathrm{MinGen}(C,u).$$

The singleton-generator orbit gives hyperedges of size one.

The current null result shows no hyperedges of size two.

Higher-width enumeration may reveal additional irreducible generating organizations.

12.19 Higher-Arity Refactoring

The present construction extends formally to arity n.

Let

$$C_n(S;U)$$

be the n-ary term-function component.

For each

$$f \in C_n,$$

compute:

$$s_S^{(n)}(f), h_S^{(n)}(f).$$

Define:

$$T_n(S) = \sum_{f \in C_n} s_S^{(n)}(f),$$
$$S_{\max,n}(S) = \max_{f \in C_n} s_S^{(n)}(f),$$

and corresponding height statistics.

A multi-arity objective may be

$$T_{\leq r}(S) = \sum_{n=1}^{r} \alpha_n T_n(S),$$

where

$$\alpha_n \geq 0$$

weights each arity.

A signature optimal at arity two may be poor at arity three.

Thus higher-arity analysis may change both preservation and optimality.

The computational obstacle is the rapid growth of

$$q^{q^n}.$$

Sparse, symbolic, orbit-based, or theorem-guided methods will be required beyond very small cases.

12.20 Sparse Closure Methods

Dense representation stores every function table explicitly.

For larger carriers or arities, a sparse approach should store only functions actually reached.

Let

$$R_k$$

be the reached set after $k$closure stages.

A sparse algorithm maintains:

hash-indexed function tables,
incremental constructor queues,
symmetry-normalized representatives,

and

checkpointed frontier states.

The memory cost scales with

$$| C_n |$$

rather than the full ambient space

$$q^{q^n}.$$

Sparse closure does not remove worst-case combinatorial explosion. It avoids allocating the entire ambient operation universe when the generated component is small.

The release includes explicit failure behavior for infeasible dense order-four configurations. Future work should make sparse execution the primary path.

12.21 Symmetry Reduction

Carrier permutations, argument permutations, and polarity actions can reduce candidate duplication.

Let a finite group Gact on the operation space.

Instead of evaluating every operation independently, one may evaluate one representative per orbit:

$$\mathcal{O}_A^{(n)}/G.$$

For any invariant computation I,

$$I(f) = I(g \cdot f)$$

after the corresponding relabeling of output records.

Exact profile invariance under carrier conjugacy justifies this reduction.

Polarity transformations require more care because they preserve generated content but may not preserve cost.

Thus symmetry reduction may be used for:

carrier relabeling,

while polarity orbits remain objects to be measured rather than automatically collapsed.

12.22 Proof Extraction

The finite algorithms establish exact values by exhaustive search.

For broader mathematical significance, computational results should be converted into human-readable theorems.

Possible proof-extraction targets include:

characterization of all singleton generators,
closed formulas for preserving-pair counts,
classification of profile types,

proof of cooperative-pair impossibility,

and

symbolic derivation of optimal break-even thresholds.

The current work already provides one example.

After establishing that there are exactly eight singleton generators and no cooperative pairs, the preserving-pair count follows from

$$| \mathcal{R}_2 | = \binom{| C |}{2} - \binom{| C | - 8}{2}.$$

A future proof should explain structurally why the singleton-generator set equals the polarity orbit and why no two nongenerator principal subcomponents join to the top.

Such a proof would replace a finite census with a family-level theorem.

12.23 Formal Verification

The strongest certification path would translate key definitions and proofs into a proof assistant.

The formal objects include:

finite carriers,
operation tables,
term evaluation,
binary closure,
minimum-cost recursion,
generator predicates,

and

landscape optimality.

A proof assistant could verify:

termination,
correctness of profile computation,
component preservation,
pair-count formulas,

and

penalized-optimality certificates.

The finite tables may be imported as data and checked by reflection.

Formal verification would reduce dependence on trust in the Python implementation.

It would not replace the need for claim discipline or literature review.

12.24 Automated Literature Equivalence

Historical classification remains one of the least automated parts of Algebraic Abiogenesis.

A discovered operation or identity family may be equivalent to known mathematics under:

symbol renaming,
argument permutation,
duality,
term equivalence,
definitional extension,

or

quotient interpretation.

Keyword search alone may miss such equivalence.

A stronger literature-audit system would generate canonical structural signatures:

identity bases,
finite-model counts,
free-algebra growth,
clone invariants,
congruence properties,

and

categorical relationships.

These signatures could be compared against mathematical databases and formal libraries.

Until such tools mature, historical novelty remains a human-supervised judgment.

12.25 Preregistered Benchmarks

A stronger empirical evaluation should preregister:

carrier orders,
signatures,
random seeds,
term bounds,
filters,
scoring rules,
baseline methods,

and

success criteria.

Relevant baselines include:

random operation search,
collision-only ranking,
fixed-operation identity mining,
direct equation enumeration,

and

known-theory retrieval.

The benchmark should ask whether operation-first formation improves:

recovery rate,
false-positive rejection,
structural diversity,
compression,

or

novelty yield

relative to those baselines.

The current release supplies calibration evidence and exact finite landscapes. It does not yet supply a large preregistered comparative benchmark.

12.26 Novelty Standard

A strong future novelty claim should survive at least the following sequence:

cross-order replication,
explanation subtraction,
adversarial countermodel search,
structural certification,
formal proof,
independent computational reproduction,

and

specialist literature review.

A candidate should also exhibit significance beyond a small isolated table.

Possible forms include:

an infinite model family,
a normal-form theorem,
a new clone-theoretic invariant,
a new minimal-generator phenomenon,
a new decision procedure,

or

a reusable discovery theorem.

The current release demonstrates methodology and finite structure. It does not claim that this stronger novelty standard has been met.

12.27 Toward a General Refactoring Functor

Let $\mathbf{Comp}_2$ be a category whose objects are finite pointed binary components and whose morphisms are isomorphisms preserving the distinguished unary operation.

Associate to each (C, u)its weighted preserving-signature landscape

$$\mathbf{R}_k(C, u).$$

An isomorphism

$$\varphi\colon (C, u) \to (D, v)$$

induces a bijection between preserving signatures:

$$S \longmapsto \varphi_*(S),$$

preserving:

$$|\ S\ |,$$
$$T(S),$$
$$S_{\max}(S),$$
$$H_{\max}(S),$$

and the complete normalized profiles.

Thus one expects a functorial assignment

$$\mathcal{R}_k\colon \mathbf{Comp}_2 \longrightarrow \mathbf{WGraph},$$

where **WGraph**is a suitable category of finite weighted graphs.

The exact categorical formulation requires choices about:

edge orientation,
weight preservation,
profile labels,

and

cost-policy transformations.

The finite isomorphism invariance established here provides the base case.

A general refactoring functor would place signature landscapes within ordinary structural mathematics rather than treating them only as computational outputs.

12.28 Toward Variational Objecthood

The optimization

$$S^* \in \arg\min_{S \in \mathcal{R}(C,u)} [T(S) + \lambda(|\ S\ | - 1)]$$

selects a primitive presentation while holding extensional content fixed.

This is an instance of a broader variational principle:

choose the least costly representation preserving declared invariants.

The preserved invariant is

$$C_2(S; u) = C.$$

The optimized variables are the primitive signatures.

The cost combines:

description complexity

and

access complexity.

This parallels variational objecthood, in which an equivalence policy is selected to maximize compression while preserving task-relevant distinctions.

Here the quotient object has already formed. The variational problem acts on its language.

Thus the two principles occupy successive levels:

optimal identity policy ⟶ optimal primitive presentation.

A complete theory of mathematical formation may require both.

12.29 What Has Been Demonstrated

The Stage 2 release demonstrates:

blind recovery of known finite algebraic families,
exact rejection of structurally defective false births,
complete finite singleton-generator spectra,
independent minimum-size and minimum-height profiles,
recursive conservative primitive promotion,
complete width-two preserving-signature landscapes,
shortest preserving refactoring paths,
absence of cooperative pairs in three principal components,

and

exact primitive-price phase transitions.

The strongest structural equation is

$$C_2(S;u) = C$$

together with the optimization

$$S_\lambda^* \in \arg\min_S \left[\sum_{f \in C} s_S(f) + \lambda(|\, S \,| - 1) \right].$$

The first fixes algebraic content.

The second selects language.

12.30 What Has Not Been Demonstrated

The release does not demonstrate:

a historically new major algebra,
a new Boolean or Kleene variety,
full clone equality across all arities,
irreducible cooperative generation,
large-carrier scalability,
fully autonomous interpretation,
fully automated novelty certification,

formal proof-assistant verification,

or

superiority over every conventional automated-discovery baseline.

It also does not establish that operation-node size is the uniquely correct measure of mathematical accessibility.

These omissions are not hidden residual defects. They define the boundary of the present result.

12.31 The Next Decisive Result

The most direct next advance is a strictly pair-born component:

$$\mathcal{R}_1(C, u) = \emptyset,$$
$$\mathcal{R}_2(C, u) \neq \emptyset.$$

A second route is a theorem proving that a broad class of pointed components cannot possess cooperative pairs.

A third route is a higher-arity result in which two binary-component-equivalent signatures separate at arity three, or are proved fully clone-equivalent.

A fourth route is target-free discovery of a useful circuit family from anonymous gates without supplying the target logical identities.

A fifth route is a novel invariant derived from the full weighted refactoring landscape and proved to distinguish algebraic components not separated by standard invariants.

These advances would move the program beyond the exact optimization of familiar finite components.

12.32 Conclusion

Algebraic Abiogenesis begins with anonymous finite operations and asks how stable laws, quotients, and theories can emerge.

The present stage continues the process after theory birth.

A formed binary component admits multiple primitive presentations. Those presentations preserve extensional content while changing exact access costs. Their complete bounded ecology can be enumerated, weighted, compared, and optimized.

The principal sequence is

finite operation ⟶ identity ⟶ theory ⟶ generator spectrum ⟶ refactoring landscape ⟶ policy-optimal language.

The calculations establish that:

same algebraic content ⇒/same generative accessibility.

They further establish that the best width-two presentations in the principal examples arise through macro enrichment, not irreducible cooperation.

The theory's language can therefore be optimized exactly, but it has not yet been shown to generate a new minimal ontology through interaction among incomplete primitives.

The current achievement is a reproducible post-birth mathematics of algebraic presentation.

The next achievement must be stronger:

not merely a cheaper language for a known component,

but

a new generative phenomenon exposed by the search itself.

13. References

1. Birkhoff, G. “On the Structure of Abstract Algebras.” Proceedings of the Cambridge Philosophical Society, vol. 31, 1935, pp. 433–454.
2. Burris, S., and H. P. Sankappanavar. A Course in Universal Algebra. Springer-Verlag, 1981.
3. McKenzie, R., G. McNulty, and W. Taylor. Algebras, Lattices, Varieties. Vol. 1, Wadsworth & Brooks/Cole, 1987.
4. Cohn, P. M. Universal Algebra. 2nd ed., D. Reidel, 1981.
5. Post, E. L. The Two-Valued Iterative Systems of Mathematical Logic. Annals of Mathematics Studies, no. 5, Princeton University Press, 1941.
6. Lau, D. Function Algebras on Finite Sets: A Basic Course on Many-Valued Logic and Clone Theory. Springer, 2006.
7. Pöschel, R., and L. A. Kalužnin. Funktionen- und Relationenalgebren: Ein Kapitel der diskreten Mathematik. Birkhäuser, 1979.
8. Szendrei, Á. Clones in Universal Algebra. Séminaire de Mathématiques Supérieures, vol. 99, Presses de l’Université de Montréal, 1986.
9. Sheffer, H. M. “A Set of Five Independent Postulates for Boolean Algebras, with Application to Logical Constants.” Transactions of the American Mathematical Society, vol. 14, no. 4, 1913, pp. 481–488.
10. Shannon, C. E. “The Synthesis of Two-Terminal Switching Circuits.” Bell System Technical Journal, vol. 28, no. 1, 1949, pp. 59–98.
11. Quine, W. V. “The Problem of Simplifying Truth Functions.” American Mathematical Monthly, vol. 59, no. 8, 1952, pp. 521–531.
12. McCluskey, E. J. “Minimization of Boolean Functions.” Bell System Technical Journal, vol. 35, no. 6, 1956, pp. 1417–1444.
13. Wegener, I. The Complexity of Boolean Functions. Wiley-Teubner, 1987.
14. Jukna, S. Boolean Function Complexity: Advances and Frontiers. Springer, 2012.
15. Kleene, S. C. Introduction to Metamathematics. North-Holland, 1952.
16. Priest, G. An Introduction to Non-Classical Logic: From If to Is. 2nd ed., Cambridge University Press, 2008.

17. Bergmann, M. An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems. Cambridge University Press, 2008.

18. Imai, Y., and K. Iséki. “On Axiom Systems of Propositional Calculi. XIV.” Proceedings of the Japan Academy, vol. 42, 1966, pp. 19–22.

19. Baader, F., and T. Nipkow. Term Rewriting and All That. Cambridge University Press, 1998.

20. Knuth, D. E., and P. B. Bendix. “Simple Word Problems in Universal Algebras.” Computational Problems in Abstract Algebra, edited by J. Leech, Pergamon Press, 1970, pp. 263–297.

21. Dershowitz, N., and J.-P. Jouannaud. “Rewrite Systems.” Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics, edited by J. van Leeuwen, Elsevier, 1990, pp. 243–320.

22. Dijkstra, E. W. “A Note on Two Problems in Connexion with Graphs.” Numerische Mathematik, vol. 1, 1959, pp. 269–271.

23. Gallo, G., G. Longo, S. Pallottino, and S. Nguyen. “Directed Hypergraphs and Applications.” Discrete Applied Mathematics, vol. 42, nos. 2–3, 1993, pp. 177–201.

24. Nemhauser, G. L., L. A. Wolsey, and M. L. Fisher. “An Analysis of Approximations for Maximizing Submodular Set Functions—I.” Mathematical Programming, vol. 14, 1978, pp. 265–294.

25. Rissanen, J. “Modeling by Shortest Data Description.” Automatica, vol. 14, no. 5, 1978, pp. 465–471.

26. Grünwald, P. D. The Minimum Description Length Principle. MIT Press, 2007.

27. Lenat, D. B. “Automated Theory Formation in Mathematics.” Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 1977, pp. 833–842.

28. Colton, S. Automated Theory Formation in Pure Mathematics. Springer, 2002.

29. McCune, W. “Solution of the Robbins Problem.” Journal of Automated Reasoning, vol. 19, no. 3, 1997, pp. 263–276.

30. Fajtlowicz, S. “On Conjectures of Graffiti.” Discrete Mathematics, vol. 72, nos. 1–3, 1988, pp. 113–118.

31. Bundy, A. “The Use of Explicit Plans to Guide Inductive Proofs.” Proceedings of the Ninth International Conference on Automated Deduction, Springer, 1988, pp. 111–120.

32. Schmidt, M., and H. Lipson. “Distilling Free-Form Natural Laws from Experimental Data.” Science, vol. 324, no. 5923, 2009, pp. 81–85.

33. Urban, J., and J. Vyskočil. “Theorem Proving in Large Formal Mathematics as an Emerging AI Field.” Automated Reasoning and Mathematics: Essays in Memory of William W. McCune, edited by M. P. Bonacina and M. E. Stickel, Springer, 2013, pp. 240–257.

34. Blanchette, J. C., S. Böhme, and L. C. Paulson. “Extending Sledgehammer with SMT Solvers.” Journal of Automated Reasoning, vol. 51, no. 1, 2013, pp. 109–128.

35. Youvan, D. C. Mathematical It from Bit: Algebraic Abiogenesis, Anonymous Finite Operations, and the Automated Birth of Abstract Algebra. July 2026. DOI: 10.13140/RG.2.2.25987.26403.

36. Youvan, D. C. How Mathematics Can Be Born: A Public Guide to Algebraic Abiogenesis and Mathematical It from Bit. July 11, 2026. ResearchGate publication 408956917. The paper describes itself as the nontechnical companion to the Stage I technical work and identifies the associated public repository.

37. Youvan, D. C. No-Corpus Artificial Intelligence from Algebraic Abiogenesis: Policy-Governed Object Formation, Theory Ecologies, and Certified Memory. July 2026. DOI: 10.13140/RG.2.2.12146.06088.

38. Youvan, D. C. Algebraic Abiogenesis Methodology and Reproducibility Repository: Stage I. GitHub, 2026. https://github.com/DougYouvan/algebraic-abiogenesis.

39. Youvan, D. C. Algebraic Abiogenesis Stage 2: Exact Signature Refactoring, Full-Generator Spectra, Recursive Promotion, and Weighted Presentation Landscapes. Experimental release v0.11.1. GitHub, 2026. https://github.com/DougYouvan/algebraic-abiogenesis-stage2.

40. OpenAI. GPT-5.6 Thinking. Computational and editorial collaboration in the preparation, execution audit, mathematical organization, and Journal Format presentation of this work. OpenAI, 2026.

The author has published over 5,700 AI-assisted papers at: https://www.researchgate.net/profile/Douglas-Youvan/research . Google Scholar has indexed these preprints, and if you want a particular reference, the AI mode of a Google search (Gemini) has efficiently catalogued these preprints.